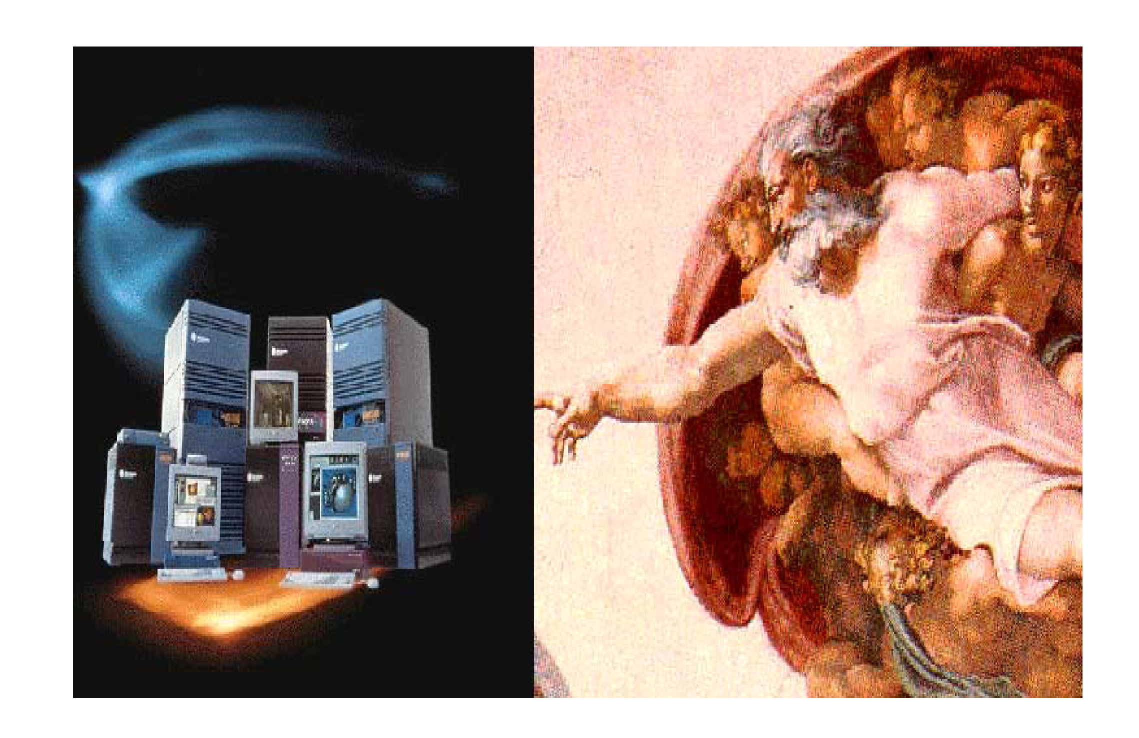
<u>SISTEMAS OPERATIVOS</u>



MAGISTER DAVID LUIS LA RED MARTINEZ

SISTEMAS

OPERATIVOS

Magister David Luis la Red Martínez

PROFESOR TITULAR POR CONCURSO DE

"SISTEMAS OPERATIVOS"

Licenciatura en Sistemas de Información

Departamento de Informática

UNIVERSIDAD NACIONAL DEL NORDESTE

U.N.N.E. - ARGENTINA

A mi familia.

A mis profesores.

A mis alumnos.

Prólogo

Es para mi una gran satisfacción prologar este libro del Profesor La Red Martínez sobre el aprendizaje de los Sistemas Operativos. El libro no es más que una muestra y un testimonio escrito de un largo proceso de formación del profesorado y de renovación en las formas docentes, que ha tenido lugar en la Universidad Nacional del Nordeste en Corrientes en el Departamento de Informática, con motivo de la Maestría en Informática y Computación, y en el que el Profesor La Red ha colaborado de una forma destacada, pudiendo decirse que no hubiera sido posible sin su participación.

El tema de los sistemas operativos es complejo e implica muchos conceptos nuevos para el alumno, algunos de ellos difíciles de comprender. Con este libro el aprendizaje se hace mucho más llevadero, e incluso divertido. El libro es fruto, por un lado, de la experiencia diaria del aula, acumulada tras muchos años de trabajo y esfuerzo, y, por otro, de la incorporación de las nuevas tecnologías a la enseñanza. Ello se nota en el estilo del libro, en la selección de los ejemplos y problemas que se incluyen, en los contenidos, y en la forma de comunicarlo al lector.

La incorporación de los recursos y herramientas informáticas de trabajo es uno de sus muchos aciertos. Sorprende la lista de ellas que han sido utilizadas. El poder acceder a multitud de programas de cálculo y diseño, dibujo, tratamiento de textos, correo electrónico, Internet, etc., desde el propio libro, en su versión electrónica, brindan al alumno la posibilidad de un aprendizaje mucho más profundo y práctico. Permiten también la auto-formación.

El Profesor La Red al poner a disposición de cualquiera el libro en Internet, comparte con todos su trabajo y conocimiento. Nada hay más grato que compartir, sabiendo que muchos, en cualquier parte del mundo, se beneficiarán de este trabajo. Esto podrá comprobarlo cuando empiece a recibir agradecimientos y consultas de los lugares más insospechados. Aunque, en la inmensa mayoría de los casos, no recibirá ni siquiera las gracias, conviene que sepa que muchos le agradecerán, en silencio, su esfuerzo y generosidad.

Por ello, mediante este libro, el Profesor La Red pasa de ser un profesor universitario a un profesor universal.

Dr. Enrique Castillo Ron.
Académico de Número de la
Academia de Ingeniería de España.
Santander (España); Noviembre de 2001.

Prefacio

Este libro trata sobre los aspectos fundamentales referidos a los Sistemas Operativos y divide el estudio de los mismos en tres partes:

- *Sistemas Operativos Convencionales.*
- *Sistemas Operativos Distribuidos.*
- *Casos de Estudio.*

Para aquellos lectores que solo deseen adquirir o refrescar conocimientos relacionados con los Sistemas Operativos en general, será suficiente con la lectura de la Primer Parte, en tanto que para aquellos que deseen un conocimiento más profundo, teniendo presente la problemática de los Sistemas Distribuidos, será necesario avanzar en la lectura del primer grupo de temas de la Segunda Parte; asimismo, si además se desea incursionar en aspectos complementarios pero importantes, se sugiere también la lectura del segundo grupo de temas de la mencionada Segunda Parte, la que se recomienda leer aunque no se tenga interés en los Sistemas Distribuidos, ya que es un buen complemento de la Primer Parte de esta obra.

En cuanto a la Tercer Parte, corresponde aclarar que resultaría de interés para quienes deseen profundizar en el estudio teórico - práctico de un conjunto de problemáticas relacionadas con los Sistemas Operativos, con el auxilio de herramientas de avanzada tales como Mathematica, Matlab, Java, Redes Neuronales, Sistemas Expertos, Orientación a Objetos, etc., siendo un buen complemento de las dos partes anteriores, que permite incrementar el conocimiento de los temas considerados.

Es preciso señalar además que este libro está destinado a los alumnos de una Carrera de Grado en Informática que deban hacer un Curso de Sistemas Operativos, pudiendo ser de utilidad, según la profundidad del curso, la Primer Parte, la Segunda Parte o las tres que componen este trabajo.

Master David Luis la Red Martínez.
Prof. Titular de la UNNE.
Corrientes (Argentina); Noviembre de 2001.

Índice General

Índice de Figuras

Índice de Tablas

Parte I

Sistemas Operativos Convencionales

Capítulo 1

Introducción

1.1 Qué es un Sistema Operativo

Una de las *definiciones* más comúnmente aceptadas expresa:

- **"Un S. O. es un grupo de programas de proceso con las rutinas de control necesarias para mantener continuamente operativos dichos programas".**

El *objetivo primario* de un Sistema Operativo es:

- Optimizar todos los recursos del sistema para soportar los requerimientos.

A los efectos de situar a los S. O. en el conjunto del software para computadoras, podemos clasificar a este de la siguiente manera:

- *Programas de sistema*:
  - Controlan la operación de la computadora en sí.
- *Programas de aplicación*:
  - Resuelven problemas para los usuarios.

En este contexto, el Sistema Operativo es el programa fundamental de todos los programas de sistema.

El S. O. protege y libera a los programadores de la complejidad del hardware, colocándose un nivel de software por sobre el hardware para:

- Controlar todas las partes del sistema.
- Presentar al usuario una interfaz o máquina virtual.

El *esquema típico de un sistema de cómputos* incluye:

- Programas de aplicación:

    - Sistema bancario, reservaciones en una línea aérea, juegos, etc.

- Programas de sistema:
    - Compiladores, editores, intérpretes de comandos.
    - Sistema Operativo.

- Hardware:
    - Lenguaje de máquina.
    - Microprogramación.
    - Dispositivos físicos

Las principales características del **microprograma** son:

- Se trata de software que generalmente se localiza en la memoria de solo lectura.
- Busca las instrucciones de lenguaje de máquina para ejecutarlas como una serie de pequeños pasos.
- El conjunto de instrucciones que interpreta define al **lenguaje de máquina.**
- En ciertas máquinas se implanta en el hardware y no es en realidad una capa distinta.

Respecto del **lenguaje de máquina** es preciso señalar que:

- Generalmente posee entre 50 y 300 instrucciones, sirviendo la mayoría para desplazar datos, hacer operaciones aritméticas y comparar valores.
- Los dispositivos de e / s (entrada / salida) se controlan al cargar valores en registros del dispositivo especiales.

Una de las **principales funciones del S. O.** es ocultar toda esta complejidad y brindar al programador un conjunto mas conveniente de instrucciones para trabajar.

El S. O. se ejecuta en modo central o modo de supervisión, con máxima prioridad y generalmente con protección por hardware.

Los compiladores, editores y demás programas se ejecutan en modo usuario.

El S. O. es la serie de programas, dispuestos ya sea en el software o en la memoria fija (microcódigo), que hacen al hardware utilizable.

Los S. O. ponen el *"poder computacional básico"* del hardware convenientemente a disposición del usuario, pero *consumen parte de ese poder computacional* para funcionar [7, Deitel].

Los S. O. son, en primer lugar, **administradores de recursos**, siendo el recurso primario el hardware del sistema.[1]

Las **principales características** de los S. O. son:

- Definir la "Interfaz del Usuario".

[1] Ver Figura 1.1 de la página 6.

- Compartir el hardware entre usuarios.
- Permitir a los usuarios compartir los datos entre ellos.
- Planificar recursos entre usuarios.
- Facilitar la entrada / salida.
- Recuperarse de los errores.

Los **principales recursos administrados** por los S. O. son:

- Procesadores.
- Almacenamiento.
- Dispositivos de e / s.
- Datos.

Los S. O. son una **interfaz** con:

- Operadores.
- Programadores de aplicaciones.
- Programadores de sistemas (administradores del S. O.).
- Programas.
- Hardware.
- Usuarios.

El S. O. debe presentar al usuario el equivalente de una **máquina extendida** o **máquina virtual** que sea mas fácil de programar que el hardware subyacente.

1.2 Historia de los Sistemas Operativos - Generaciones

Los S. O. han estado relacionados históricamente con la arquitectura de las computadoras en las cuales se ejecutan, razón por la cual su historia puede analizarse según las siguientes generaciones y sus principales características [7, Deitel]:

- *Generación Cero* (década de 1940):
  - Carencia total de S. O.
  - Completo acceso al lenguaje de máquina.
- *Primera generación* (1945-1955): bulbos y conexiones:
  - Carencia de S. O.

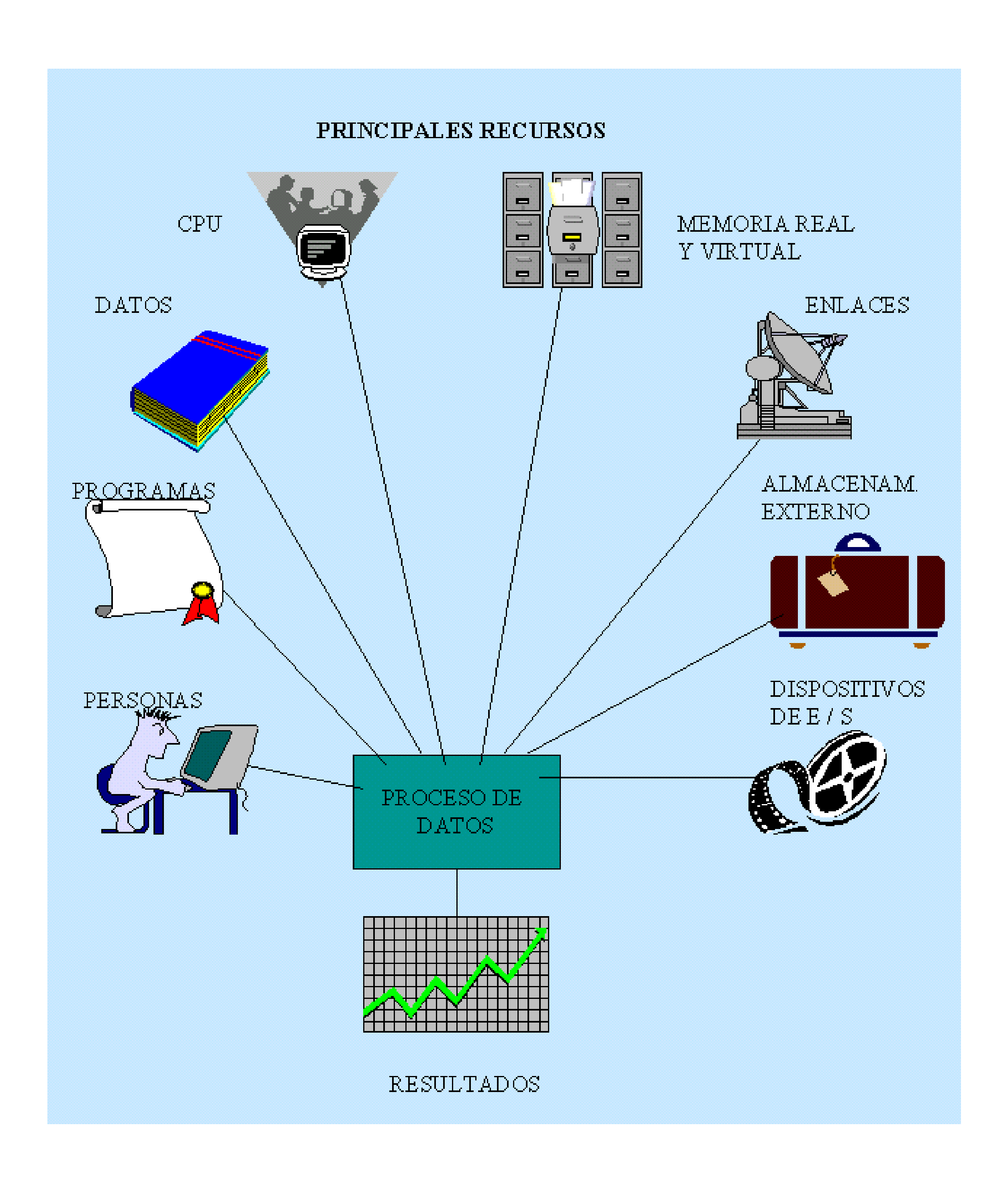


Figura 1.1: Recursos administrados por el S. O.

  - En los años cincuenta comienzan como transición entre trabajos, haciendo la misma más simple.

- *Segunda generación* (1955-1965): transistores y sistemas de procesamiento por lotes (batch):

  - En los años sesenta aparecen los S. O. para sistemas compartidos con:
    * **Multiprogramación:** varios programas de usuarios se encuentran al mismo tiempo en el almacenamiento principal, cambiando el procesador rápidamente de un trabajo a otro.
    * **Multiprocesamiento:** varios procesadores se utilizan en un mismo sistema para incrementar el poder de procesamiento.
  - Posteriormente aparece la **independencia de dispositivo:**
    * El programa del usuario especifica las características de los dispositivos que requieren los archivos.
    * El S. O. asigna los dispositivos correspondientes según los requerimientos y las disponibilidades.

- *Tercera generación* (1965-1980): circuitos integrados y multiprogramación:

  - Difusión de la **multiprogramación:**
    * Partición de la memoria en porciones, con trabajos distintos en cada una de ellas.
    * Aprovechamiento del tiempo de espera consecuencia de operaciones de e / s, para utilizar la CPU para otros procesos.
  - Protección por hardware del contenido de cada partición de memoria.
  - Aparición de técnicas de **spooling:**
    * Simultaneous Peripheral Operation On Line: operación simultánea y en línea de periféricos.
    * Almacenamiento de trabajos de entrada y de salida en dispositivos transitorios rápidos (discos), para disminuir el impacto de los periféricos mas lentos.
  - Son **sistemas de modos múltiples**, es decir que deben soportar **sistemas de propósitos generales;** son grandes y complejos pero muy poderosos.
  - Interponen una **capa de software** entre el usuario y el hardware.
  - Aparecen los **lenguajes de control de trabajos,** necesarios para especificar el trabajo y los recursos requeridos.
  - Soportan **timesharing (tiempo compartido),** variante de la multiprogramación con usuarios conectados mediante terminales en línea, permitiendo la operación en **modo interactivo** o **conversacional.**
  - Aparecen los **sistemas de tiempo real,** que requieren tiempos de respuesta muy exigentes, especialmente para usos industriales o militares.

- Se difunden las computadoras de rango medio.

- *Cuarta generación* (1980-1990): computadoras personales:
  - Aparición de software **amigable con el usuario**, destinado a usuarios no profesionales y con una interfase gráfica muy desarrollada.
  - Desarrollo de **sistemas operativos de red** y **sistemas operativos distribuidos.**
  - *Sistemas operativos de red*:
    * Los usuarios están conscientes de la existencia de varias computadoras conectadas.
    * Cada máquina ejecuta su propio S. O. local.
    * Son similares a los S. O. de un solo procesador pero con el agregado de:
      · Controlador de interfaz de la red y su software de bajo nivel.
      · Software para conexión y acceso a archivos remotos, etc.
  - *Sistemas operativos distribuidos*:
    * Aparece ante los usuarios como un S. O. de un solo procesador, aún cuando de soporte a varios procesadores.
    * Los usuarios no son conscientes del lugar donde se ejecutan sus programas o donde se encuentran sus archivos, ya que lo debe administrar el S. O. automáticamente.
    * Deben permitir que un programa se ejecute mediante varios procesadores a la vez, maximizando el paralelismo.
  - Aparición de emuladores de terminal para el acceso a equipos remotos desde computadoras personales (PC).
  - Gran énfasis en la **seguridad**, en especial por el desarrollo de los sistemas de comunicaciones de datos.
  - El S. O. crea un ambiente de trabajo según el concepto de **máquina virtual**, que lo aísla del funcionamiento interno de la máquina.
  - Proliferación de **sistemas de bases de datos**, accesibles mediante redes de comunicación.

1.3 Conceptos de los Sistemas Operativos

La interfaz entre el S. O. y los programas del usuario se define como el conjunto de **"instrucciones ampliadas"** [23, Tanenbaum] que proporciona el S. O. y son las **"llamadas al sistema":**

  - Crean, eliminan y utilizan **objetos del software** controlados por el S. O.:
    * Los mas importantes son **procesos** y **archivos.**

- **Procesos:**

- Es el concepto central de todos los S. O.
- Es básicamente un programa en ejecución.
- Consta del programa ejecutable, sus datos y pila, contador y otros registros, además de la información necesaria para ejecutar el programa.
- La información de control relacionada con los procesos se almacena en la **tabla de procesos:**
  - Es administrada por el S. O.
  - Posee un arreglo de estructuras, una por cada proceso existente en ese momento.
- Un proceso (suspendido) consta de:
  - Un espacio de dirección.
  - Los datos pertinentes de la tabla de procesos.
- Un proceso puede crear **procesos hijo** y estos nuevos procesos hijo, conformando un **árbol de procesos.**

• **<u>Archivos:</u>**

- Una de las funciones principales del S. O. es brindar **independencia de dispositivo**.
- Muchos S. O. soportan el concepto de **directorio** como una forma de agrupar archivos.
- Los directorios se estructuran jerárquicamente, por lo que a cada archivo le corresponde una **ruta de acceso.**
- Existen distintos esquemas de seguridad de archivos en los distintos S. O.

• **<u>Llamadas al sistema:</u>**

- Permiten a los programas comunicarse con el S. O. y solicitarle servicios.
- A cada llamada le corresponde un procedimiento:
  - Pone los parámetros de la llamada en un lugar específico para luego ejecutar una instrucción tipo "trap" de llamada a procedimiento protegido para iniciar el S. O.
  - Luego de "trap" el S. O. recupera el control , examina los parámetros y si son válidos ejecuta el trabajo solicitado.
  - Luego de terminar, el S. O. coloca un código de estado en un registro indicando si tuvo éxito o fracaso y ejecuta una instrucción del tipo "return from trap" para regresar el control al procedimiento.
  - El procedimiento regresa al programa llamador con un código de estado como un valor de función; dentro de los parámetros pueden regresar valores adicionales.

1.4 Estructura de los Sistemas Operativos

Se considera la **organización interna** de los S. O. y conforme a ella se los clasifica de la siguiente manera, destacándose sus principales características:

- **Sistemas monolíticos:**
  - Es muy común: no existe estructura propiamente dicha o es mínima.
  - El S. O. es una colección de procedimientos que se pueden llamar entre sí.[2]
  - Cada procedimiento tiene una interfaz bien definida en términos de parámetros y resultados.
  - Para ejecutar los servicios del S. O. (llamadas al sistema):[3]
    - Se solicitan colocando los parámetros en lugares bien definidos (registros o pilas).
    - Se ejecuta una instrucción especial de trampa: **llamada al núcleo** o **llamada al supervisor.**
    - La instrucción cambia la máquina del **modo usuario** al **modo núcleo (**o **modo supervisor).** [23, Tanenbaum]
    - Se transfiere el control al S. O.
    - El S. O. examina los parámetros de la llamada para determinar cuál de ellas se desea realizar.
    - El S. O. analiza una tabla que contiene en la entrada *"k"* un apuntador al procedimiento que realiza la *"k-ésima"* llamada al sistema:
      - Identifica al procedimiento de servicio llamado.
    - La llamada al sistema termina y el control regresa al programa del usuario.

- **Sistemas con capas:**
  - Es una generalización del modelo de estructura simple para un sistema monolítico.
  - Consiste en organizar el s. o. como una jerarquía de capas, cada una construida sobre la inmediata inferior.
  - El primer sistema con este esquema fue el "THE" (Holanda - Dijkstra -1968):[4]
    - *"THE": Technische Hogeschool Eindhoven.*
    - Capa 0:
      - Trabaja con la asignación del procesador.
      - Alterna entre los procesos cuando ocurren las interrupciones o expiran los cronómetros.

[2] Ver Figura 1.2 de la página 11 [23, Tanenbaum].
[3] Ver Figura 1.3 de la página 11 [23, Tanenbaum].
[4] Ver Tabla 1.1 de la página 12 [23, Tanenbaum].

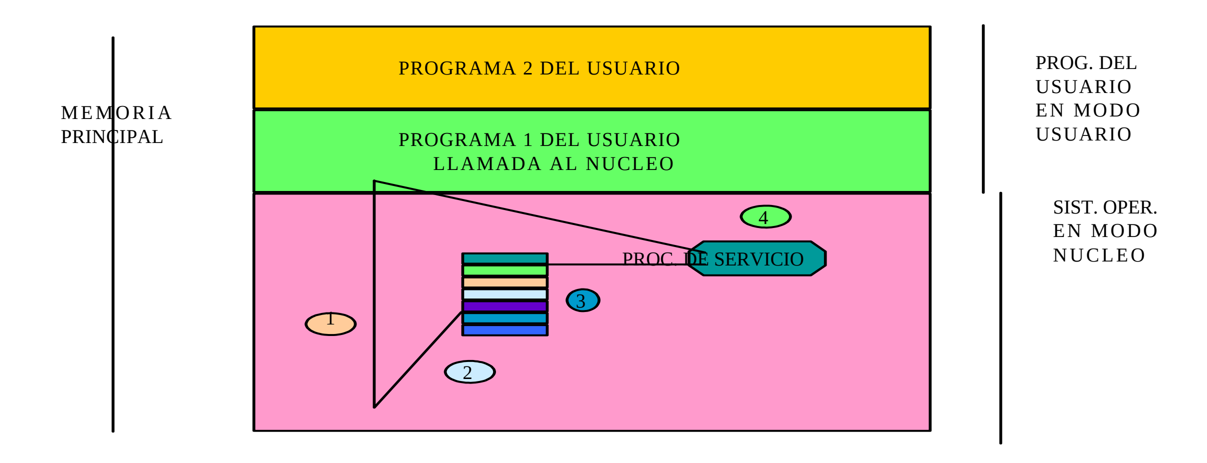


Figura 1.2: Modelo de estructura simple para un sistema monolítico.

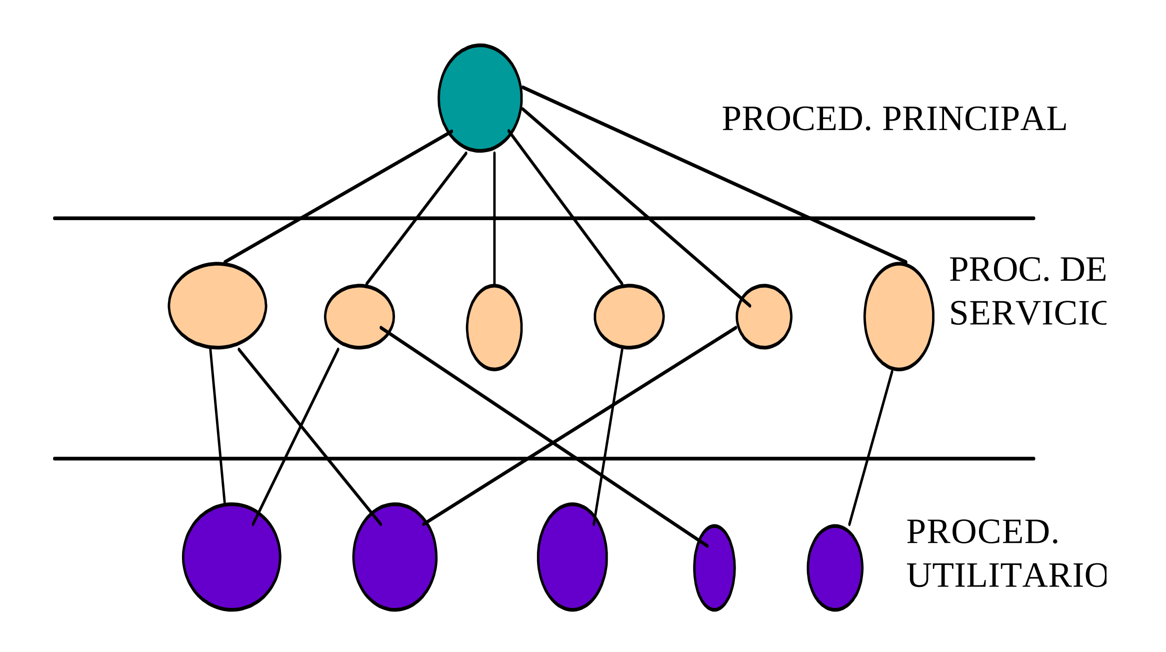


Figura 1.3: Forma de llamada al sistema en un sistema monolítico.

5 - Operador
4 - Programas del Usuario
3 - Control de Entrada / Salida
2 - Comunicaciones Operador - Proceso
1 - Administración de la Memoria y del Disco
0 - Asignación del Procesador y Multiprogramación

Tabla 1.1: Estructura del S.O. en capas "THE".

- · Proporciona la multiprogramación básica.

- ∗ Capa 1:
  - · Administra la memoria.
  - · Asegura que las páginas (porciones de memoria) requeridas de los procesos lleguen a memoria cuando fueran necesarias.
- ∗ Capa 2:
  - · Administra la comunicación entre cada proceso y la consola del operador.
  - · Por sobre esta capa, cada proceso tiene su propia consola de operador.
- ∗ Capa 3:
  - · Controla los dispositivos de e / s y almacena en buffers los flujos de información entre ellos.
  - · Por sobre la capa 3 cada proceso puede trabajar con dispositivos abstractos de e / s en vez de con dispositivos reales.
- ∗ Capa 4:
  - · Aloja los programas del usuario.
  - · Los programas. del usuario no tienen que preocuparse por el proceso, memoria, consola o control de e / s.
- ∗ Capa 5:
  - · Localiza el proceso operador del sistema.

- – Una generalización mas avanzada del concepto de capas se presento con "Multics" (MIT, Bell Labs y General Electric):
  - ∗ *"Multics": multiplexed information and computing service.*
  - ∗ Presenta una estructura en anillos concéntricos, siendo los interiores los privilegiados.
  - ∗ Un procedimiento de un anillo exterior, para llamar a un procedimiento de un anillo interior, debe hacer el equivalente a una llamada al sistema.

- **Máquinas virtuales:**
  - – Se separan totalmente las funciones de multiprogramación y de máquina extendida.

- Existe un elemento central llamado **monitor de la máquina virtual** que:
  * Se ejecuta en el hardware.
  * Realiza la multiprogramación.
  * Proporciona varias máquinas virtuales a la capa superior.
- Las máquinas virtuales instrumentan copias "exactas" del hardware simple, con su modo núcleo / usuario, e / s, interrupciones y todo lo demás que posee una máquina real.
- Pueden ejecutar cualquier S. O. que se ejecute en forma directa sobre el hardware.
- Las distintas máquinas virtuales pueden ejecutar distintos S. O. y en general así lo hacen.
- Soportan periféricos virtuales.
- Ejemplo de S. O. representativo de esta estructura: "VM/370" de IBM:[5]
  * Las m. v. generalmente utilizaran, entre otros, el S. O. "CMS": Conversational Monitor System.
  * Cuando un programa "CMS" ejecuta una llamada al sistema:
    · La llamada es atrapada por el S. O. en su propia m. v.; no pasa directamente al "VM/370".
    · "CMS" proporciona las instrucciones de e / s en hardware para la lectura del disco virtual o lo necesario para efectuar la llamada.
    · "VM/370" atrapa estas instrucciones de e / s y las ejecuta sobre el hardware verdadero.

• **Modelo cliente - servidor:**

- Una tendencia en los S. O. modernos es la de explotar la idea de mover el código a capas superiores y mantener un **núcleo mínimo,** de manera similar al "VM/370".
- Implantar la mayoría de las funciones del S. O. en los procesos del usuario.
- Para solicitar un servicio (por ej.: lectura de un bloque de cierto archivo) según el modelo cliente - servidor:[6]
  * El proceso del usuario (**proceso cliente**) envía la solicitud a un **proceso servidor:**
    · Realiza el trabajo y regresa la respuesta.
- El núcleo controla la comunicación entre los clientes y los servidores.
- Se fracciona el S. O. en partes, cada una controlando una faceta:
  * Servicio a archivos, a procesos, a terminales, a memoria, etc., cada parte pequeña y más fácilmente controlable.

[5]Ver Figura 1.4 de la página 14 [23, Tanenbaum].
[6]Ver Figura 1.5 de la página 15 [23, Tanenbaum].

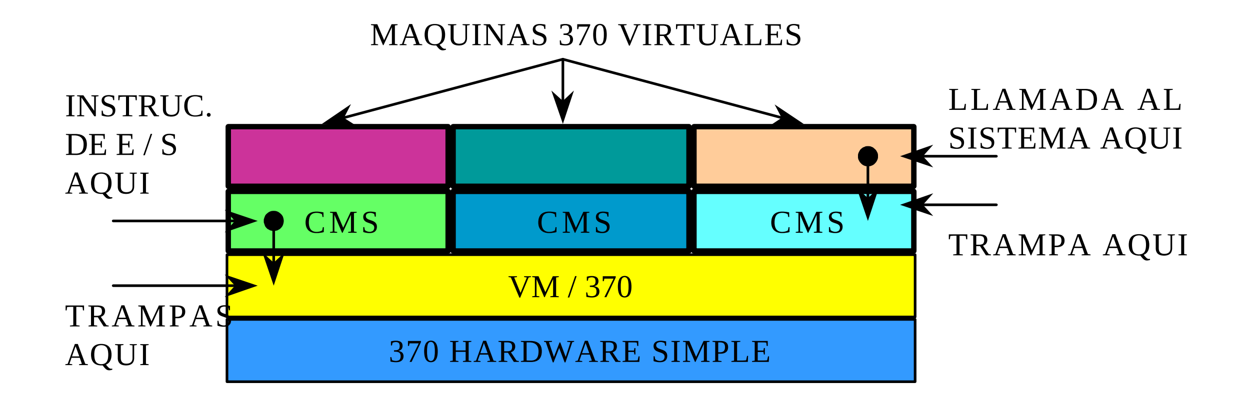


Figura 1.4: La estructura de VM/370 con CMS.

- Los servidores se ejecutan como procesos en modo usuario:
  - No tienen acceso directo al hardware.
  - Se aíslan y acotan más fácilmente los problemas.
- Se adapta para su uso en los sistemas distribuidos:[7]
  - Si un cliente se comunica con un servidor mediante mensajes:
    - No necesita saber si el mensaje se atiende localmente o mediante un servidor remoto, situado en otra máquina conectada.
    - Envía una solicitud y obtiene una respuesta.
- Algunas funciones del S. O., por ej. el cargado de comandos en los registros físicos del dispositivo de e / s, presentan problemas especiales y distintas soluciones:
  - Ejecución en modo núcleo, con acceso total al hardware y comunicación con los demás procesos mediante el mecanismo normal de mensajes.
  - Construcción de un mínimo de **mecanismos** dentro del núcleo manteniendo las decisiones de **política** relativas a los usuarios dentro del espacio del usuario.

1.5 Tendencias

Las principales tendencias en S. O. son las siguientes [7, Deitel]:

- Soporte generalizado para multiprocesamiento.
- Migración hacia el microcódigo de funciones de los S. O. realizadas por software.
- Distribución del control entre procesadores localizados.
- Mejora de la eficiencia en el soporte de la ejecución concurrente de programas.

[7] Ver Figura 1.6 de la página 15 [23, Tanenbaum].

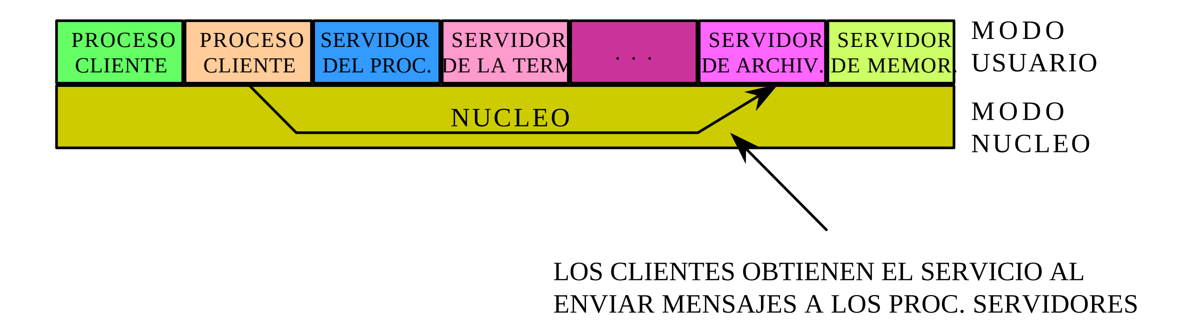


Figura 1.5: El modelo cliente - servidor.

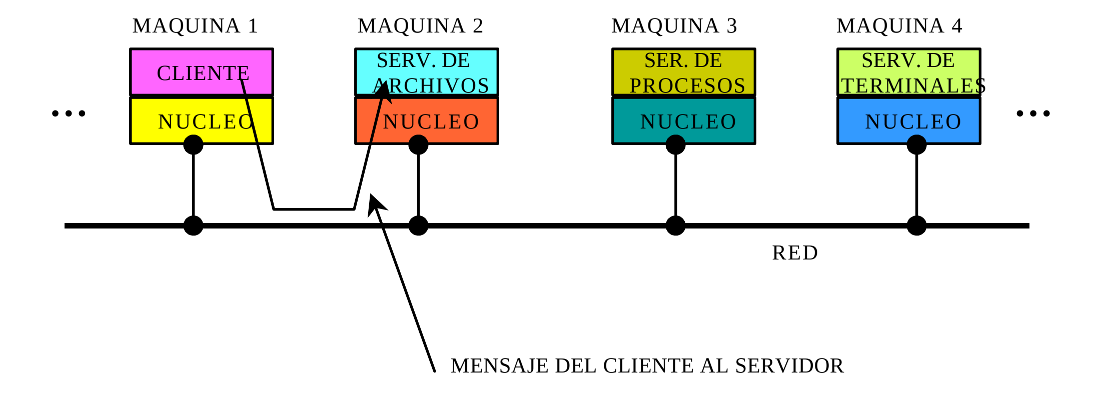


Figura 1.6: El modelo cliente - servidor en un sistema distribuido.

- Soporte del paralelismo masivo con altísimo grado de concurrencia.
- Profundización de los esquemas de máquinas virtuales.
- Continuación del esquema de familias de S. O. para familias de computadoras, viendo las aplicaciones *máquinas virtuales*.
- Compatibilidad con nuevas generaciones de computadoras.
- Desarrollos en la ingeniería de software para brindar S. O. más preservables, confiables y comprensibles.
- Proliferación de redes de sistemas, distribuyendo tareas en equipos sobre los que el usuario puede no tener conocimiento ni control con énfasis en la importancia de la perspectiva de las máquinas virtuales.
- Permanencia del concepto de almacenamiento virtual.
- Permanencia de la perspectiva del S. O. como administrador de recursos, teniendo presente que los datos serán considerados cada vez más como un recurso para ser administrado.
- Profundización del desarrollo de S. O. con funciones distribuidas entre varios procesadores a través de grandes redes de sistemas [26, Tanenbaum].

1.6 Hardware

Los principales aspectos del hardware, de importancia para los S. O., son los siguientes [7, Deitel]:

- **Compaginación del almacenamiento:**
  - Objetivo: acelerar el acceso al almacenamiento primario (bancos de memoria).
  - Generalmente, mientras cualquiera de las localidades de un **banco de almacenamiento primario,** está siendo accedida, ninguna otra referencia puede estar en curso.
  - La compaginación del almacenamiento coloca localidades de memoria adyacentes en diferentes bancos de almacenamiento, para permitir varias referencias al mismo tiempo.
- **Registro de relocalización:**
  - Permite relocalizar de forma dinámica los programas.
  - La dirección base de un programa en la memoria principal se sitúa en el registro de relocalización.
  - El contenido del registro de relocalización se añade a cada dirección desarrollada por un programa en ejecución.

  - Permite al programa residir en localizaciones diferentes a aquellas para las cuales fue traducido.

- **Interrupciones y escrutinio:**

  - *Interrupciones*: permiten a una unidad obtener la inmediata atención de otra, de manera que la primera pueda informar de un cambio de estado:
    - Permite salvar el "estado" de la unidad interrumpida antes de procesar la interrupción.
  - *Escrutinio*: técnica que permite que una unidad verifique el estado de otra unidad de funcionamiento independiente.

- **Utilización del "buffer":**

  - Un "buffer" es un área de almacenamiento primario destinada a contener datos durante transferencias de e / s.
  - Cuando concluye la transferencia los datos pueden ser accedidos por el procesador.
  - Esquema de "*entradas de buffer simple*":
    - El canal deposita datos en el buffer.
    - El procesador procesa estos datos.
    - El canal deposita nuevos datos, etc.
    - No puede haber simultaneidad entre operaciones de colocar datos en el buffer y procesarlos:
      - Afecta la performance.
  - Esquema de "*entradas de buffer doble*":
    - Permite la sobreposición de operaciones de e / s con el procesamiento:
      - Mejora la performance.
    - Mientras el canal deposita datos en un buffer el procesador puede estar procesando los datos del otro buffer.
    - Cuando el procesador concluye el proceso de los datos del primer buffer, puede continuar con los datos del segundo, mientras el canal deposita nuevos datos en el primer buffer:
    - Es la técnica de "*buffer biestable (*o *en flip flop)*".

- **Dispositivos periféricos:**

  - Permiten el almacenamiento de grandes cantidades de información fuera del almacenamiento principal.
  - Existen dispositivos secuenciales y de acceso directo.
  - Las características y prestaciones son muy variadas.

- **Protección del almacenamiento:**

- Limita el número de direcciones que un programa puede referenciar.
- Es esencial en los sistemas multiusuario.
- Se implementa mediante los "*registros de límites*", que definen las direcciones superior e inferior del bloque de almacenamiento afectado a un determinado programa.
- También se pueden utilizar "*claves de protección del almacenamiento*" anexas a áreas de almacenamiento primario:
  - Un programa solo puede acceder a localidades de almacenamiento cuyas claves de protección concuerdan con las del programa.

- **Temporizadores y relojes:**
  - "*Temporizador de intervalos*": previene que un solo usuario monopolice el procesador en sistemas multiusuario.
  - El temporizador genera una interrupción al procesador cuando expira el intervalo asignado a un usuario.
  - "*Reloj horario*": permite al computador hacer un seguimiento de la "*hora del reloj de pared*", con una exactitud de millonésimas de segundo o mayor.

- **Operaciones en línea y fuera de línea; procesadores satélite:**
  - "*Operación en línea*": los periféricos utilizados están conectados al procesador.
  - "*Operación fuera de línea*": los periféricos utilizados están conectados a unidades de control que no están conectadas al sistema central o principal.

- **Canales de entrada / salida:**
  - Son sistemas computacionales de propósito especial, dedicados al manejo de la e / s con independencia del procesador principal.
  - Tienen acceso directo al almacenamiento principal para almacenar o recuperar información.
  - Evitan al procesador la mayor parte de la carga de manejar la e / s, incrementando la concurrencia.
  - Los principales tipos de canales son los siguientes:
    - Selectores.
    - Multiplexores de bytes.
    - Multiplexores de bloques.

- **Robo de ciclo:**
  - Significa que en la *competencia entre el procesador y los canales* para acceder a un determinado banco de almacenamiento primario (memoria principal), *se da prioridad a los canales*:

    * Se optimiza el uso de los dispositivos de e / s.

- **Direccionamiento de base más desplazamiento:**
  - Todas las direcciones son añadidas al contenido de un "*registro de base*".
  - Los programas son "*independientes de la localización*":
    * Especialmente importante en ambientes multiusuario.

- **Estado de problema, estado supervisor, instrucciones privilegiadas:**
  - Corresponde a distintos "*estados de ejecución*".
  - "*Estado de problema o de usuario*": estado en que corren los programas de usuario:
    * Tiene acceso a un subconjunto de instrucciones del conjunto de instrucciones de la máquina.
  - "*Estado supervisor o de núcleo*": generalmente el S. O. corre así con la categoría de "*usuario de mayor confianza o nivel*":
    * Tiene acceso a todas las instrucciones del conjunto de instrucciones de la máquina.
  - Si el sistema soporta más de dos estados:
    * Se puede instrumentar una "*granulación de protección*" más fina.
    * Permite conceder accesos por medio del "*principio de menos privilegio*":
      · Se debe garantizar a cada usuario en particular la menor cantidad de privilegio y acceso que necesite para cumplir sus tareas.
  - "*Instrucciones privilegiadas*": son aquellas a las que no se tiene acceso en estado de problema.

- **Almacenamiento virtual:**
  - Los sistemas de almacenamiento virtual permiten a los programas referenciar direcciones que no necesitan corresponder con las direcciones reales disponibles en el almacenamiento primario.
  - Las "*direcciones virtuales*" desarrolladas por los programas en ejecución son traducidas dinámicamente por el hardware a las "*direcciones reales*" de instrucciones y datos del almacenamiento principal.
  - Los programas pueden referenciar espacios de direcciones mucho mayores que los espacios de direcciones disponibles en el almacenamiento primario.
  - Se utilizan técnicas de:
    * "*Paginación*": bloques de datos de tamaño fijo van o vienen entre el almacenamiento primario y el secundario.
    * "*Segmentación*": identifica las unidades lógicas de los programas y datos para facilitar el control de acceso y participación.

- **Multiprocesamiento:**
  - Varios procesadores comparten un almacenamiento primario común y un solo S. O.
  - Es necesario "secuencializar" el acceso a una localización (dirección) de almacenamiento compartido para que dos o más procesadores no intenten:
    - Modificarla al mismo tiempo.
    - Modificarla uno(s) mientras otro(s) intenta(n) leerla.

- **Acceso directo a la memoria (DMA):**
  - Requiere una sola interrupción al procesador por cada bloque de caracteres transferidos durante la operación de e / s, lo cual mejora significativamente la performance (rendimiento).
  - Es como si el procesador, en vez de interrumpido fuera retrasado.
  - Resulta muy útil para altos requerimientos de e / s.
  - "*Canal DMA*": es el hardware responsable del robo de ciclos y de la operación de los dispositivos de e / s.

- **Canalización:**
  - Técnica de hardware utilizada para explotar ciertos tipos de paralelismo durante el procesamiento de instrucciones.
  - Varias instrucciones pueden estar simultáneamente en diferentes estados de ejecución.

- **Jerarquía de almacenamiento:**
  - Los niveles de almacenamiento incluyen:
    - Almacenamiento primario: memoria principal.
    - Almacenamiento secundario: discos, cintas, etc.
    - Almacenamiento "*caché*": memoria muy veloz diseñada para aumentar la velocidad de ejecución de los programas:
      - Aloja la parte (instrucciones y datos) en ejecución de un programa.
  - Los niveles de almacenamiento crean "*jerarquías de almacenamiento*": caché, almacenamiento primario, almacenamiento secundario.
  - Al bajar en la jerarquía:
    - Descienden el costo y la velocidad.
    - Aumenta la capacidad.
  - "*Espacio de direcciones*": conjunto de todas las direcciones disponibles para un programa.

1.7 Software

Consiste en los programas de instrucciones y datos que definen para el hardware los algoritmos necesarios para la resolución de problemas.

Los aspectos más destacados en relación con los S. O. son los siguientes [7, Deitel]:

- **Programación en lenguaje de máquina:**
  - "*Lenguaje de máquina*":
    - Lenguaje de programación que un computador puede comprender directamente.
    - Es "*dependiente de la máquina*": un programa en lenguaje de máquina escrito en el computador de un fabricante, generalmente no puede ser ejecutado en el de otro, salvo que su lenguaje de máquina sea compatible.
    - Muy poco usado actualmente.
- **Ensambladores y macroprocesadores:**
  - Los "*lenguajes ensambladores*" se desarrollaron para:
    - Incrementar la velocidad de programación .
    - Reducir los errores de codificación.
  - Los programas deben ser traducidos al "*lenguaje de máquina*" mediante un programa "*ensamblador*":
    - También es dependiente de la máquina.
  - Los "*macroprocesadores*":
    - Se desarrollaron para acelerar la codificación de un programa ensamblador.
    - Se incorporaron en los ensambladores.
    - Una "*macroinstrucción*" indica la ejecución de varias instrucciones en lenguaje ensamblador.
    - El "*procesador de macroinstrucciones*" efectúa una "*macroexpansión*" cuando lee una macro durante la traducción de un programa:
      - Genera una serie de instrucciones en lenguaje ensamblador correspondientes a la macro.
- **Compiladores:**
  - "*Lenguajes de alto nivel*": se desarrollaron para resolver el problema de la dependencia respecto a la máquina.
  - Permiten el desarrollo de programas "*independientes de la máquina*".
  - Se logra mayor velocidad de programación, programas transportables entre sistemas diferentes y menores requerimientos de conocimientos de hardware.
  - "*Compiladores*": traducen los lenguajes de alto nivel al lenguaje de máquina.
  - "*Traductores*": es la denominación para "*compiladores*" y "*ensambladores*".

    * Entrada: "*programa fuente*" del programador.
    * Salida: "*programa objeto*" o "*programa resultante*".

- **Sistemas de control de entrada / salida (IOCS: input / output control system):**

  - EL IOCS libera al programador de aplicaciones de la complejidad de la administración de la e / s:
    * Programas de canal, coordinación de canales y procesadores, control de la e / s, etc.
  - Es una manifestación de la tendencia a que los desarrolladores de aplicaciones se concentren en la producción de códigos orientados hacia las aplicaciones y no hacia los sistemas (hardware).

- **Utilización del SPOOL (Simultaneous Peripheral Operation On Line: operación simultánea de periféricos en línea):**

  - Un dispositivo de alta velocidad (ej.: disco) se interpone entre un programa en ejecución y un dispositivo de baja velocidad (ej.: impresora) relacionado con el programa en la e / s.
  - Evita la demora en la ejecución de programas como consecuencia del uso de periféricos lentos.

- **Lenguajes orientados hacia el procedimiento versus lenguajes orientados hacia el problema:**

  - "*O. hacia el procedimiento*": son de propósito general y aptos para resolver gran variedad de problemas:
    * Ej.: Pascal, Cobol, Fortran, Basic, PL/I.
  - "*O. hacia el problema*": son específicos para resolver determinados tipos de problemas:
    * Ej.: GPSS (simulación), SPSS (estadística).

- **Compiladores rápidos y sucios versus compiladores optimizadores:**

  - "*C. rápidos y sucios*": producen rápidamente un programa objeto que puede ser ineficiente respecto de almacenamiento y velocidad de ejecución:
    * Utiles para el desarrollo y prueba de sistemas.
  - "*C. optimizadores*": producen con mayor lentitud un código de máquina altamente eficiente en almacenamiento y ejecución:
    * Utiles en etapa de producción de los sistemas.

- **Interpretadores:**

  - No producen un programa objeto.

- Ejecutan directamente un programa fuente.
- Son útiles en ambientes de desarrollo de programas.
- Son más lentos que los códigos compilados.

- **Cargadores absolutos y de relocalización:**
  - Los programas se ejecutan en el almacenamiento principal.
  - "*Asignación*": es la asociación de instrucciones y datos con localizaciones particulares de almacenamiento.
  - "*Cargador*": es un programa que coloca las instrucciones y datos de un programa dentro de localizaciones del almacenamiento principal.
  - "*Cargador absoluto*": coloca las instrucciones y datos en las localizaciones específicas indicadas en el programa de lenguaje de máquina.
  - "*Cargador de relocalización*": puede cargar un programa en varios lugares dentro del almacenamiento principal:
    - Depende de la disponibilidad de almacenamiento primario al momento de realizar la carga.
  - "*Tiempo de carga*": momento de realizar la carga.

- **Cargadores de enlace y editores de enlace:**
  - El programa en lenguaje de máquina producido por un traductor debe ser combinado con otros programas en lenguaje de máquina para formar una unidad ejecutable.
  - La "*combinación de programas*" es realizada por "*cargadores de enlace*" y "*editores de enlace*" antes del tiempo de ejecución del programa.
  - "*Cargador de enlace*": en el momento de carga, combina cualesquiera programas requeridos y los carga directamente en el almacenamiento primario.
  - "*Editor de enlace*": ejecuta la combinación de programas mencionada y además crea una imagen de carga a memoria que preserva en el almacenamiento secundario (disco), para usos futuros:
    - Es muy útil en ambientes de producción, ya que la carga inmediata de la imagen de memoria previamente producida evita un nuevo proceso de combinación de programas previo a cada ejecución.

1.8 Memoria Fija

El concepto de "*microprogramación*" suele atribuirse al prof. Maurice Wilkes (1951) [7, Deitel].

La primer aplicación a gran escala fueron los S/360 (IBM - '60) [24, Tanenbaum].

La "*microprogramación dinámica*": permite cargar fácilmente los nuevos "*microprogramas*" ("*microcódigo*") dentro del "*almacenamiento de control*", desde donde son ejecutados:

- Permite variar dinámica y frecuentemente los conjuntos de instrucciones de máquina.

La "*microprogramación*" introduce una capa de programación por debajo del lenguaje de máquina:

- Hace posible definir las instrucciones del lenguaje de máquina.

Los "*microprogramas*" están formados por "*microinstrucciones*" individuales que en relación a las instrucciones de los lenguajes de máquina son de:

- Naturaleza mucho más elemental.
- Función más dispersa.

Cada instrucción de lenguaje de máquina es implementada por un microprograma completo que puede ser extenso:

- El almacenamiento de control debe ser mucho más rápido que el almacenamiento primario.

Microcódigos vertical y horizontal:
El "*microcódigo vertical*":

- Es similar a la ejecución de instrucciones en lenguaje de máquina.
- Especifica el movimiento de uno o varios datos entre registros.

El "*microcódigo horizontal*":

- Está constituído por microinstrucciones que requieren muchos más bits.
- Puede especificar la operación paralela de movimiento de datos entre muchos o todos los registros de datos de la unidad de control.
- Es más poderoso pero más complejo que el microcódigo vertical.

Decisión de qué funciones implementar en microcódigo:
Una importante cuestión de diseño es decidir qué funciones del sistema computacional se implementarán en microcódigo.

El microcódigo permite mejorar el rendimiento en la ejecución de un sistema computacional.

El criterio frecuentemente es colocar en la memoria fija (en vez de en el software) las secuencias de instrucciones utilizadas con más frecuencia.

Emulación:
Es una técnica por medio de la cual se hace que una máquina aparente ser otra.

El conjunto de instrucciones de lenguaje de máquina que va a ser emulada se microprograma en la "*máquina anfitriona*".

Los programas de lenguaje de máquina de la máquina emulada pueden ejecutarse directamente en la anfitriona.

Es útil para compatibilidad y migración de sistemas.

Microdiagnósticos:
Los microprogramas tienen más acceso al hardware que los programas de lenguaje de máquina:

- Es posible efectuar detección y corrección de errores más amplia a un nivel más fino.

Se puede intercalar el "*microdiagnóstico*" con las instrucciones de programas de lenguaje de máquina.

Computadores personalizados:

El hardware proporciona un ambiente de propósito general para ejecutar programas de software:

- Moldean el sistema computacional según las necesidades del usuario.

En algunos sistemas los usuarios pueden efectuar esta adaptación por medio del microcódigo.

Asistencias de microcódigo:

Implementan varias rutinas de manejo de interrupciones de uso más frecuente en microcódigo a fin de lograr mejoras significativas en la ejecución.

Microprogramación y sistemas operativos:

Las funciones implementadas frecuentemente en microcódigo son las siguientes:

- Manejo de interrupciones.
- Mantenimiento de varios tipos de estructuras de datos.
- Primitivas de sincronización que controlan el acceso a los datos compartidos y otros recursos.
- Operaciones de palabras parciales que permiten que las operaciones de manipulación de bits sean manejadas en forma eficiente.
- "*Intercambio de contexto*", por ej., intercambio rápido del procesador entre los usuarios de un sistema de usuarios múltiples.
- Secuencias de regreso y llamada al procedimiento.

Capítulo 2

Procesos y Administración del Procesador

2.1 Introducción y Definiciones Sobre Procesos

El concepto central de cualquier Sistema Operativo es el de **proceso**: una abstracción de un programa en ejecución también llamada **tarea**.

No hay un acuerdo universal sobre una definición de proceso, pero sí algunas definiciones aceptadas [7, Deitel]:

- Un programa que se está ejecutando.
- Una actividad asincrónica.
- El **emplazamiento del control** de un procedimiento que está siendo ejecutado.
- Aquello que se manifiesta por la existencia en el Sistema Operativo de un **bloque de control de proceso**.
- Aquella entidad a la cual son asignados los procesadores.
- La unidad **despachable**.

En sistemas de **multiprogramación** la cpu alterna de programa en programa, en un esquema de **seudoparalelismo** , es decir que la cpu ejecuta en cierto instante un solo programa, intercambiando muy rápidamente entre uno y otro.

El **paralelismo real de hardware** se da en las siguientes situaciones:

- En ejecución de instrucciones de programa con más de un procesador de instrucciones en uso simultáneamente.
- Con la superposición de ejecución de instrucciones de programa con la ejecución de una o más operaciones de entrada / salida.

El objetivo es aumentar el paralelismo en la ejecución.

El **modelo de procesos** posee las siguientes características:

- Todo el software ejecutable, inclusive el Sistema Operativo, se organiza en varios **procesos secuenciales** o **procesos**.
- Un proceso incluye al programa en ejecución y a los valores activos del contador, registros y variables del mismo.
- Conceptualmente cada proceso tiene su propia cpu virtual.
- Si la cpu alterna entre los procesos, la velocidad a la que ejecuta un proceso no será uniforme, por lo que es necesario aclarar lo siguiente:
  - Que los procesos no deben programarse con hipótesis implícitas acerca del tiempo.
  - Que normalmente la mayoría de los procesos no son afectados por la multiprogramación subyacente de la cpu o las velocidades relativas de procesos distintos.
- Un proceso es una actividad de un cierto tipo, que tiene un programa, entrada, salida y estado.
- Un solo procesador puede ser compartido entre varios procesos con cierto "*algoritmo de planificación*" , el cual determina cuándo detener el trabajo en un proceso y dar servicio a otro distinto[1].

En cuanto a las **jerarquías de procesos** es necesario señalar que los Sistemas Operativos deben disponer de una forma de crear y destruir procesos cuando se requiera durante la operación, teniendo además presente que los procesos pueden generar procesos hijos mediante llamadas al Sistema Operativo, pudiendo darse ejecución en paralelo.

Respecto de los **estados del proceso** deben efectuarse las siguientes consideraciones:

- Cada proceso es una entidad independiente pero frecuentemente debe interactuar con otros procesos[2].
- Los procesos pueden bloquearse en su ejecución porque:
  - Desde el punto de vista lógico no puede continuar porque espera datos que aún no están disponibles.
  - El Sistema Operativo asignó la cpu a otro proceso.
- Los estados [23, Tanenbaum] que puede tener un proceso son[3]:
  - **En ejecución**: utiliza la cpu en el instante dado.
  - **Listo**: ejecutable, se detiene en forma temporal para que se ejecute otro proceso.
  - **Bloqueado**: *no se puede ejecutar* debido a la ocurrencia de algún evento externo.
- Son posibles cuatro **transiciones** entre estos estados.

[1] Ver Figura 2.1 de la página 29 [23, Tanenbaum].
[2] Ver Figura 2.2 de la página 29 [23, Tanenbaum].
[3] Ver Figura 2.3 de la página 29 [23, Tanenbaum].

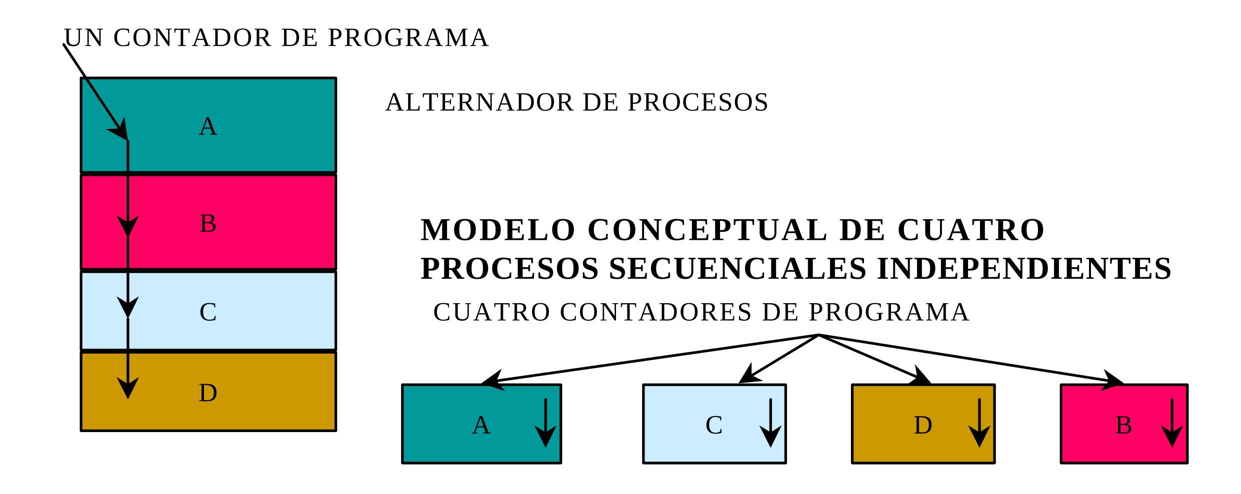


Figura 2.1: Multiprogramación de cuatro programas.

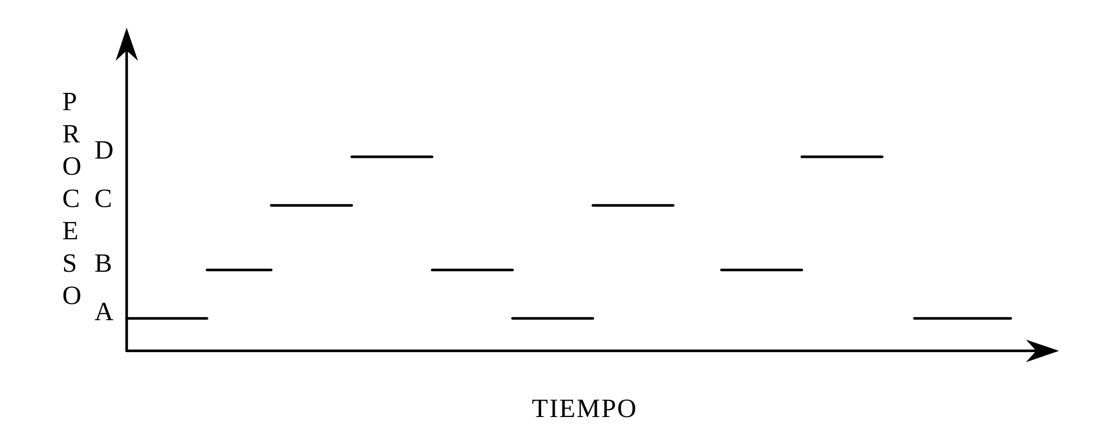


Figura 2.2: Solo un programa está activo en un momento dado.

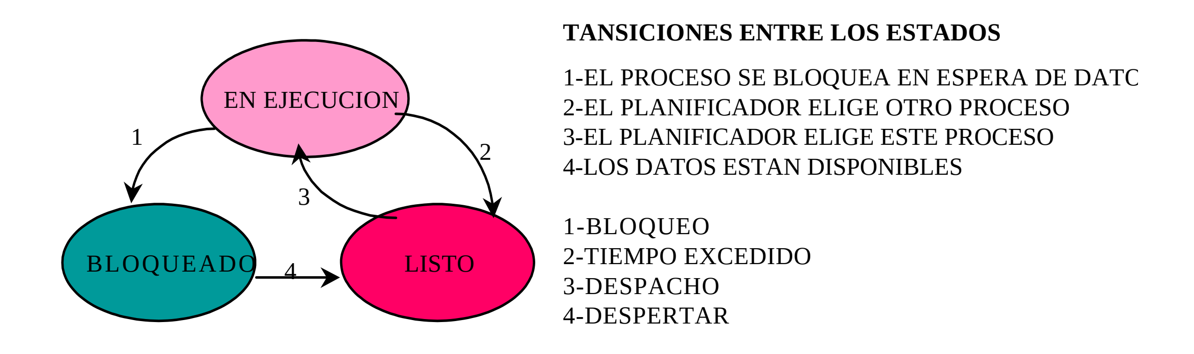


Figura 2.3: Un proceso puede estar en ejecución, bloqueado o listo.

2.2 Estados de Procesos

Durante su existencia un proceso pasa por una serie de estados discretos, siendo varias las circunstancias que pueden hacer que el mismo cambie de estado.

Debido a ello se puede establecer una "Lista de Listos" para los procesos "listos" y una "Lista de Bloqueados" para los "bloqueados".

La "*Lista de Listos*" se mantiene en orden prioritario y la "*Lista de Bloqueados*" está desordenada, ya que los procesos se desbloquean en el orden en que tienen lugar los eventos que están esperando.

Al admitirse un trabajo en el sistema se crea un proceso equivalente y es insertado en la última parte de la "*Lista de Listos*".

La asignación de la cpu al primer proceso de la "*Lista de Listos*" se denomina "Despacho" , que es ejecutado por una entidad del Sistema Operativo llamada "Despachador" .

El "Bloqueo" es la única transición de estado iniciada por el propio proceso del usuario, puesto que las otras transiciones son iniciadas por entidades ajenas al proceso.

La manifestación de un proceso en un Sistema Operativo es un "*Bloque de Control de Proceso*" (PCB) con información que incluye [7, Deitel]:

- Estado actual del proceso.
- Identificación única del proceso.
- Prioridad del proceso.
- Apuntadores para localizar la memoria del proceso.
- Apuntadores para asignar recursos.
- Area para preservar registros.

Cuando el Sistema Operativo cambia la atención de la cpu entre los procesos, utiliza las áreas de preservación del PCB para mantener la información que necesita para reiniciar el proceso cuando consiga de nuevo la cpu.

Los sistemas que administran los procesos deben poder crear, destruir, suspender, reanudar, cambiar la prioridad, bloquear, despertar y despachar un proceso.

La "*creación*" de un proceso significa:

- Dar nombre al proceso.
- Insertar un proceso en la lista del sistema de procesos conocidos.
- Determinar la prioridad inicial del proceso.
- Crear el bloque de control del proceso.
- Asignar los recursos iniciales del proceso.

Un proceso puede "*crear un nuevo proceso*", en cuyo caso el proceso creador se denomina "*proceso padre*" y el proceso creado "*proceso hijo*" y se obtiene una "*estructura jerárquica de procesos*".

La "*destrucción*" de un proceso implica:

- Borrarlo del sistema.
- Devolver sus recursos al sistema.
- Purgarlo de todas las listas o tablas del sistema.
- Borrar su bloque de control de procesos.

Un proceso "*suspendido*" no puede proseguir hasta que otro proceso lo reanude.
Reanudar (reactivar) un proceso implica reiniciarlo en el punto donde fue suspendido.

La "*destrucción*" de un proceso puede o no significar la destrucción de los procesos hijos, según el Sistema Operativo.

Generalmente se denomina "Tabla de Procesos" al conjunto de información de control sobre los distintos procesos.

2.3 Procesamiento de Interrupciones

Una "*interrupción*" es un evento que altera la secuencia en que el procesador ejecuta las instrucciones; es un hecho generado por el hardware del computador [7, Deitel].

Cuando ocurre una interrupción, el Sistema Operativo:

- Obtiene el control.
- Salva el estado del proceso interrumpido, generalmente en su bloque de control de procesos.
- Analiza la interrupción.
- Transfiere el control a la rutina apropiada para la manipulación de la interrupción.

Una interrupción puede ser iniciada por un proceso en estado de ejecución o por un evento que puede o no estar relacionado con un proceso en ejecución.

Generalmente las **interrupciones** se pueden clasificar por **tipos** según el siguiente detalle[4]:

- "SVC (llamada al supervisor)": es una petición generada por el usuario para un servicio particular del sistema, por ejemplo, realización de Entrada / Salida u obtención de más memoria.
- "Entrada / Salida": son iniciadas por el hardware de Entrada / Salida, indicando a la cpu que ha cambiado el estado de un canal o dispositivo, por ejemplo, finalización de Entrada / Salida u ocurrencia de un error.

[4] Ver Tabla 2.1 de la página 32 [7, Deitel].

Tipo de Interrupción	Descripción
SVC	Llamada al Sistema Operativo
Entrada / Salida	Cambio de estado de un canal o dispositivo
Externa	Evento externo al sistema
De Reinicio	Reinicio del procesamiento
De Verificación de Programa	Errores de procesos
De Verificación de Máquina	Errores de hardware

Tabla 2.1: Tipos de interrupciones.

- "Externas": son causadas por distintos eventos, por ejemplo, expiración de un cuanto en un reloj de interrupción o recepción de una señal de otro procesador en un sistema multiprocesador.
- "De reinicio": ocurren al presionar la "tecla de reinicio" o cuando llega una instrucción de reinicio de otro procesador en un sistema multiprocesador.
- "De verificación de programa": son causadas por errores producidos durante la ejecución de procesos, por ejemplo:
  - Un intento de dividir por cero.
  - Un intento de un proceso de usuario de ejecutar una instrucción privilegiada.
  - Un intento de ejecutar un código de operación inválido.
- "De verificación de máquina": son ocasionadas por un mal funcionamiento del hardware.

El Sistema Operativo incluye rutinas llamadas "**Manipuladores de Interrupciones (IH)**" para procesar cada tipo diferente de interrupción.

Cuando se produce una interrupción el Sistema Operativo efectúa las siguientes acciones:

- Salva el estado del proceso interrumpido.
- Dirige el control al manipulador de interrupciones adecuado.
- Se aplica la técnica de "Cambio de Contexto" .

Los Sistemas Operativos instrumentan información de control que puede aparecer como las "Palabras de Estado de Programa (PSW)", las cuales controlan el orden de ejecución de las instrucciones y contienen información sobre el estado del proceso.

Existen tres tipos de PSW, que son la "actual", la "nueva" y la "vieja".

La "PSW Actual" almacena la dirección de la próxima instrucción que será ejecutada e indica los tipos de instrucciones actualmente "habilitadas" e "inhabilitadas".

En un **sistema uniprocesador** existe:

- Solo una PSW actual.
- Seis PSW nuevas (una para cada tipo de interrupción).
- Seis PSW viejas (una para cada tipo de interrupción).

La PSW nueva para un tipo de interrupción dado contiene la dirección en el hardware donde reside el manipulador de interrupciones para este tipo específico.

Cuando ocurre una interrupción para la cual el procesador no está inhabilitado, ocurren las siguientes acciones:

- El hardware cambia las PSW en los casos siguientes:
  - Al almacenar la PSW actual en la PSW vieja, para este tipo de interrupción.
  - Al almacenar la PSW nueva en la PSW actual, para este tipo de interrupción.
- Luego de este "intercambio de PSW":
  - La PSW actual contiene la dirección del manipulador de interrupción adecuado.
  - El manipulador de interrupciones procesa la interrupción.
  - Luego de procesar la interrupción, la cpu es enviada al:
    - Proceso que estaba en ejecución en el momento de la interrupción, o al
    - Proceso de listo de más alta prioridad.
  - La acción precedente depende de si el proceso de interrupción es:
    - "**Apropiativo**" : obtiene la cpu solo si no hay procesos de listos.
    - "**No apropiativo**" : obtiene de nuevo la cpu.

2.4 El Núcleo del Sistema Operativo

El "*núcleo*" del Sistema Operativo controla todas las operaciones que implican procesos y representa solo una pequeña porción del código de todo el Sistema Operativo pero es de amplio uso [7, Deitel].

Generalmente permanece en el almacenamiento primario.

El proceso de interrupciones se incluye en el núcleo ya que debe ser rápido (especialmente en sistemas multiusuario), para optimizar el uso de los recursos del sistema y proveer tiempos de respuesta aceptables a los usuarios interactivos.

El **núcleo** inhabilita las interrupciones mientras responde a una interrupción. Las interrupciones son habilitadas de nuevo después de completar el proceso de una interrupción.

El **núcleo** del Sistema Operativo generalmente realiza las siguientes **funciones**:

- Manipulación de interrupciones.
- Creación y destrucción de procesos.
- Cambio de estados de procesos.

Criterio	Descripción
Equidad	Garantizar que cada proceso obtiene su proporción justa de la cpu
Eficacia	Mantener ocupada la cpu el ciento por ciento del tiempo
Tiempo de respuesta	Minimizar el tiempo de respuesta para los usuarios interactivos
Tiempo de regreso	Minimizar el tiempo que deben esperar los usuarios por lotes (batch) para obtener sus resultados
Rendimiento	Maximizar el número de tareas procesadas por hora

Tabla 2.2: Criterios de un buen algoritmo de planificación.

- Despacho.
- Suspensión y reanudación de procesos.
- Sincronización de procesos.
- Comunicación entre procesos.
- Manipulación de bloques de control de proceso.
- Soporte de las actividades de Entrada / Salida.
- Soporte de la asignación y desasignación de almacenamiento.
- Soporte del sistema de archivos.
- Soporte de un mecanismo de llamada / regreso al procedimiento.
- Soporte de ciertas funciones contables (estadísticas) del sistema.

2.5 Planificación de Procesos

Cuando más de un proceso es ejecutable desde el punto de vista lógico, el Sistema Operativo debe decidir cuál de ellos debe ejecutarse en primer término.

El **Planificador** es la porción del Sistema Operativo que decide y el **Algoritmo de Planificación** es el utilizado.

Los principales "criterios" respecto de un buen algoritmo de planificación [23, Tanenbaum] son la *equidad*, la *eficacia*, el *tiempo de respuesta*, el *tiempo de regreso* y el *rendimiento*[5].

Algunas de estas metas son contradictorias, por ejemplo, minimizar el tiempo de respuesta para los usuarios interactivos significaría no ejecutar las tareas batch.

[5] Ver Tabla 2.2 de la página 34 [23, Tanenbaum].

Cada proceso es único e impredecible, es decir que pueden requerir intensivamente operaciones de Entrada / Salida o intensivamente cpu; el planificador del Sistema Operativo no tiene la certeza de cuánto tiempo transcurrirá hasta que un proceso se bloquee, ya sea por una operación de Entrada / Salida o por otra razón .

Para evitar que un proceso se apropie de la cpu un tiempo excesivo, los equipos poseen un dispositivo que provoca una interrupción en forma periódica, por ejemplo 60 hz, o sea sesenta veces por segundo.

En cada interrupción del reloj el Sistema Operativo decide si el proceso que se está ejecutando continúa o si el proceso agotó su tiempo de cpu y debe suspenderse y ceder la cpu a otro proceso.

Los principales conceptos relacionados con **Planificación del Procesador** son los siguiente:

- *Planificación apropiativa* : es la estrategia de permitir que procesos ejecutables (desde el punto de vista lógico) sean suspendidos temporalmente.
- *Planificación no apropiativa* : es la estrategia de permitir la ejecución de un proceso hasta terminar.
- *Planificación del procesador* : determinar cuándo deben asignarse los procesadores y a qué procesos, lo cual es responsabilidad del Sistema Operativo.

2.6 Niveles de Planificación del Procesador

Se consideran tres niveles importantes de planificación, los que se detallan a continuación[6]:

- *Planificación de alto nivel*:
  - También se denomina *Planificación de trabajos*.
  - Determina a qué trabajos se les va a permitir competir activamente por los recursos del sistema, lo cual se denomina *Planificación de admisión*.
- *Planificación de nivel intermedio*:
  - Determina a qué procesos se les puede permitir competir por la cpu.
  - Responde a fluctuaciones a corto plazo en la carga del sistema y efectúa "suspensiones" y "activaciones" ("reanudaciones") de procesos.
  - Debe ayudar a alcanzar ciertas metas en el rendimiento total del sistema.
- *Planificación de bajo nivel*:
  - Determina a qué proceso listo se le asigna la cpu cuando esta queda disponible y asigna la cpu al mismo, es decir que "despacha" la cpu al proceso.
  - La efectúa el Despachador del Sistema Operativo, el que opera muchas veces por segundo y reside siempre en el almacenamiento primario.

Los distintos Sistemas Operativos utilizan varias Políticas de Planificación, que se instrumentan mediante Mecanismos de Planificación .

[6] Ver Figura 2.4 de la página 36 [7, Deitel].

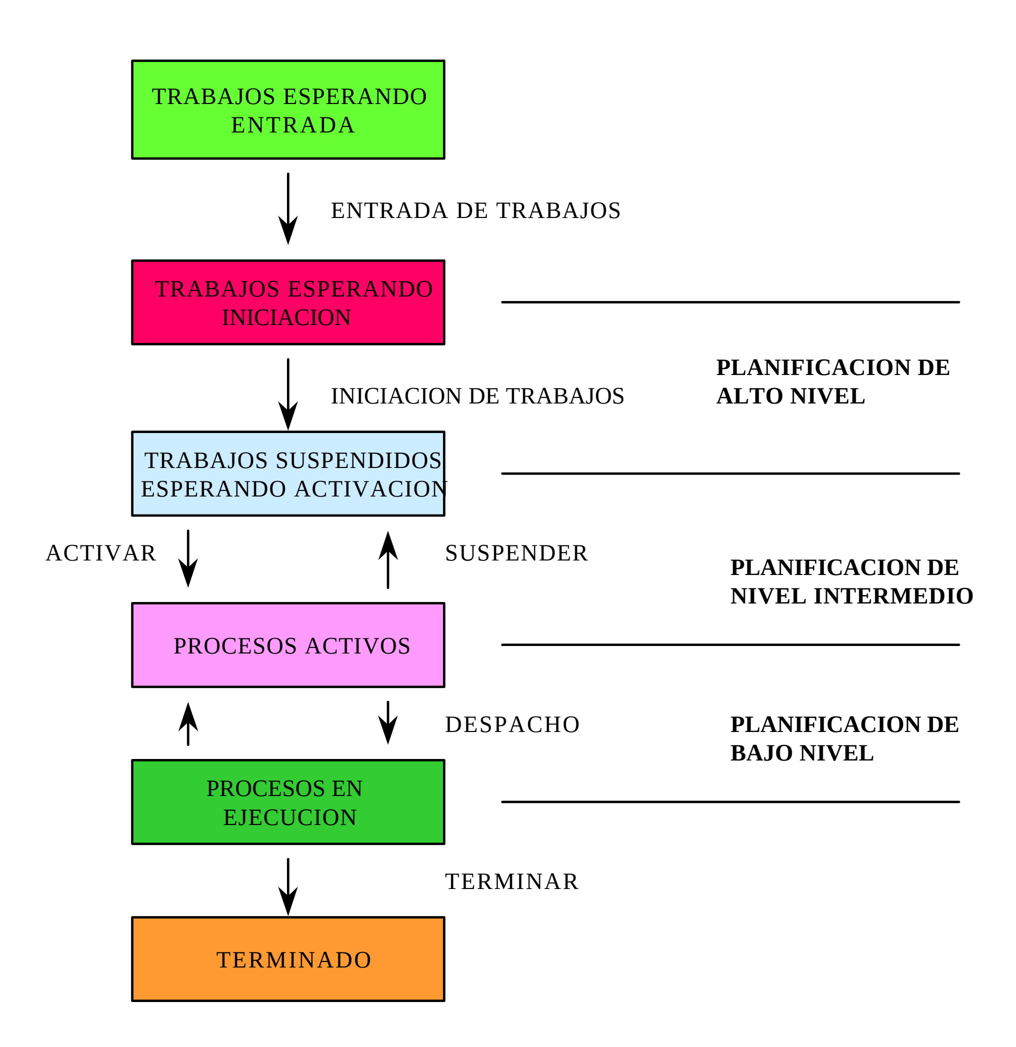


Figura 2.4: Niveles de planificación del procesador.

2.7 Objetivos de la Planificación

Los objetivos de la planificación del procesador son los siguientes e involucran a los conceptos detallados seguidamente [7, Deitel]:

- **Ser justa**:
  - Todos los procesos son tratados de igual manera.
  - Ningún proceso es postergado indefinidamente.
- **Maximizar la capacidad de ejecución**:
  - Maximizar el número de procesos servidos por unidad de tiempo.
- **Maximizar el número de usuarios interactivos que reciban unos tiempos de respuesta aceptables**:
  - En un máximo de unos segundos.
- **Ser predecible**:
  - Un trabajo dado debe ejecutarse aproximadamente en la misma cantidad de tiempo independientemente de la carga del sistema.
- **Minimizar la sobrecarga**:
  - No suele considerarse un objetivo muy importante.
- **Equilibrar el uso de recursos**:
  - Favorecer a los procesos que utilizarán recursos infrautilizados.
- **Equilibrar respuesta y utilización**:
  - La mejor manera de garantizar buenos tiempos de respuesta es disponer de los recursos suficientes cuando se necesitan, pero la utilización total de recursos podrá ser pobre.
- **Evitar la postergación indefinida**:
  - Se utiliza la estrategia del *"envejecimiento"* .
  - Mientras un proceso espera por un recurso su prioridad debe aumentar, así la prioridad llegará a ser tan alta que el proceso recibirá el recurso esperado.
- **Asegurar la prioridad**:
  - Los mecanismos de planificación deben favorecer a los procesos con prioridades más altas.

- **Dar preferencia a los procesos que mantienen recursos claves**:
  - Un proceso de baja prioridad podría mantener un recurso clave, que puede ser requerido por un proceso de más alta prioridad.
  - Si el recurso es no apropiativo, el mecanismo de planificación debe otorgar al proceso un tratamiento mejor del que le correspondería normalmente, puesto que es necesario liberar rápidamente el recurso clave.

- **Dar mejor tratamiento a los procesos que muestren un "comportamiento deseable"**:
  - Un ejemplo de comportamiento deseable es una tasa baja de paginación.

- **Degradarse suavemente con cargas pesadas**:
  - Un mecanismo de planificación no debe colapsar con el peso de una exigente carga del sistema.
  - Se debe evitar una carga excesiva mediante las siguientes acciones:
    - No permitiendo que se creen nuevos procesos cuando la carga ya es pesada.
    - Dando servicio a la carga más pesada al proporcionar un nivel moderadamente reducido de servicio a todos los procesos.

Muchas de estas metas se encuentran en conflicto entre sí, por lo que la planificación se convierte en un problema complejo.

2.8 Criterios de Planificación

Para realizar los objetivos de la planificación, un *mecanismo de planificación* debe considerar lo siguiente [7, Deitel]:

- La limitación de un proceso a las operaciones de Entrada / Salida: cuando un proceso consigue la cpu, ¿la utiliza solo brevemente antes de generar una petición de Entrada / Salida?.

- La limitación de un proceso a la cpu: cuando un proceso obtiene la cpu, ¿tiende a usarla hasta que expira su tiempo?.

- Si un proceso es por lote (batch) o interactivo: los usuarios interactivos deben recibir inmediato servicio para garantizar buenos tiempos de respuesta.

- ¿Qué urgencia tiene una respuesta rápida?: por ejemplo, un proceso de tiempo real de un sistema de control que supervise una refinería de combustible requiere una respuesta rápida, más rápida que la respuesta requerida por un proceso en lotes (batch) que deberá entregarse al día siguiente.

- La prioridad de un proceso: a mayor prioridad mejor tratamiento.

- Frecuentemente un proceso genera fallos (carencias) de página:

Disciplina	Descripción
"Apropiativa"	Una vez que se le ha otorgado la cpu a un proceso, *le puede ser retirada*
"No Apropiativa"	Una vez que se le ha otorgado la cpu a un proceso, *no le puede ser retirada*

Tabla 2.3: Disciplinas de planificación del procesador.

- Probablemente los procesos que generan pocos fallos de página hayan acumulado sus "conjuntos de trabajo" en el almacenamiento principal.
- Los procesos que experimentan gran cantidad de fallos de página aún no han establecido sus conjuntos de trabajo.
- Un criterio indica favorecer a los procesos que han establecido sus conjuntos de trabajo.
- Otro criterio indica favorecer a los procesos con una tasa alta de fallos de página ya que rápidamente generarán una petición de Entrada / Salida.

- Frecuentemente un proceso ha sido apropiado por otro de más alta prioridad, lo cual significa lo siguiente:
  - A menudo los procesos apropiados deben recibir un tratamiento menos favorable.
  - Cada vez que el Sistema Operativo asume la sobrecarga para hacer ejecutar este proceso, el corto tiempo de ejecución antes de la apropiación no justifica la sobrecarga de hacer ejecutar al proceso en primer lugar.
- ¿Cuánto tiempo de ejecución real ha recibido el proceso?: un criterio considera que debe ser favorecido un proceso que ha recibido muy poco tiempo de cpu.
- ¿Cuánto tiempo adicional va a necesitar el proceso para terminar?: los tiempos promedio de espera pueden reducirse priorizando los procesos que requieren de un tiempo de ejecución mínimo para su terminación, pero pocas veces es posible conocer la cantidad de tiempo adicional que cada proceso necesita para terminar.

2.9 Planificación Apropiativa Versus No Apropiativa

Las *Disciplinas de Planificación* pueden ser **Apropiativas** o **No Apropiativas**[7].

Las principales características de la **planificación apropiativa** son las siguientes:

- Es útil cuando los procesos de alta prioridad requieren atención rápida.
- Es importante para garantizar buenos tiempos de respuesta en sistemas interactivos de tiempo compartido.

[7] Ver Tabla 2.3 de la página 39 [23, Tanenbaum].

- Tiene su costo en recursos, ya que el intercambio de contexto implica sobrecarga y además requiere mantener muchos procesos en el almacenamiento principal, en espera de la cpu, lo que también implica sobrecarga.

Las principales características de la **planificación no apropiativa** son las siguientes:

- Significa que los trabajos "largos" hacen esperar a los trabajos "cortos".

- Logra más equidad en el tratamiento de los procesos.

- Logra hacer más predecibles los tiempos de respuesta puesto que los trabajos nuevos de prioridad alta no pueden desplazar a los trabajos en espera.

El diseño de un mecanismo apropiativo hace necesario considerar las arbitrariedades de casi cualquier esquema de prioridades, en razón de que muchas veces las propias prioridades no son asignadas de forma significativa [25, Tanenbaum].

El mecanismo debería ser sencillo pero efectivo y significativo.

2.10 Temporizador de Intervalos o Reloj de Interrupción

El proceso al cual está asignada la cpu se dice que está en ejecución y puede ser un proceso de Sistema Operativo o de usuario.

El Sistema Operativo dispone de mecanismos para quitarle la cpu a un proceso de usuario para evitar que monopolice el sistema.

El Sistema Operativo posee un "*reloj de interrupción*" o "*temporizador de intervalos*" para generar una interrupción, en algún tiempo futuro específico o después de un transcurso de tiempo en el futuro; la cpu es entonces despachada hacia el siguiente proceso [7, Deitel].

Un proceso retiene el control de la cpu hasta que ocurra alguna de las siguientes situaciones:

- La libera voluntariamente.

- El reloj la interrumpe.

- Alguna otra interrupción atrae la atención de la cpu.

Si el reloj interrumpe un proceso de usuario, la interrupción causa la ejecución del Sistema Operativo, el que decide cuál será el proceso que obtendrá la cpu.

El reloj de interrupción ayuda a garantizar tiempos de respuesta razonables a usuarios interactivos, ya que evita que el sistema se "cuelgue" a un solo usuario en un ciclo infinito y permite que los procesos respondan a "eventos dependientes del tiempo".

Asimismo, los procesos que necesitan una ejecución periódica dependen del reloj de interrupción [22, Tanenbaum].

Tipos de prioridades
Asignadas automáticamente por el sistema
Asignadas desde el exterior
Dinámicas
Estáticas
Asignadas racionalmente
Asignadas arbitrariamente

Tabla 2.4: Tipos de prioridades.

2.11 Prioridades

Las **prioridades** pueden ser de distinto tipo[8].

En el caso de prioridades asignadas arbitrariamente, un mecanismo del sistema necesita distinguir entre procesos sin importarle cuál es el más importante.

Las principales características de las **prioridades estáticas** son las siguientes:

- No cambian.
- Los mecanismos de implementación son sencillos.
- Implican una sobrecarga relativamente baja.
- No responden a cambios en el ambiente (contexto) que harían deseable ajustar alguna prioridad.

Las principales características de las **prioridades dinámicas** son las siguientes:

- Responden al cambio.
- La prioridad inicial asignada a un proceso puede durar poco tiempo, luego se la reajusta a un mejor valor.
- Los mecanismos de implementación son más complicados que para prioridades estáticas.
- Implican una sobrecarga mayor que para esquemas estáticos.

Respecto de las **prioridades adquiridas**, se hace referencia al tratamiento especial que en situaciones excepcionales requiere un cierto proceso, lo que puede significar restar recursos a los demás procesos.

[8] Ver Tabla 2.4 de la página 41 [7, Deitel].

2.12 Tipos de Planificación

2.12.1 Planificación a Plazo Fijo

Ciertos trabajos se planifican para ser terminados en un tiempo específico o plazo fijo. Es una planificación compleja debido a los siguientes factores:

- El usuario debe suministrar anticipadamente una lista precisa de recursos necesarios para el proceso, pero generalmente no se dispone de dicha información.
- La ejecución del trabajo de plazo fijo no debe producir una grave degradación del servicio a otros usuarios.
- El sistema debe planificar cuidadosamente sus necesidades de recursos hasta el plazo fijo, lo que se puede complicar con las demandas de recursos de nuevos procesos que ingresen al sistema.
- La concurrencia de varios procesos de plazo fijo (activos a la vez) puede requerir métodos sofisticados de optimización.
- La administración intensiva de recursos puede generar una considerable sobrecarga adicional.

2.12.2 Planificación Garantizada

Se establecen compromisos de desempeño con el proceso del usuario, por ejemplo, si existen "n" procesos en el sistema, el proceso del usuario recibirá cerca del "$1/n$" de la potencia de la cpu.

El sistema debe tener un registro del tiempo de cpu que cada proceso ha tenido desde su entrada al sistema y del tiempo transcurrido desde esa entrada.

Con los datos anteriores y el registro de procesos en curso de ejecución, el sistema calcula y determina qué procesos están más alejados por defecto de la relación "$1/n$" prometida y prioriza los procesos que han recibido menos cpu de la prometida.

2.12.3 Planificación del Primero en Entrar Primero en Salir (FIFO)

Es muy simple, los procesos se despachan de acuerdo con su tiempo de llegada a la cola de listos.

Una vez que el proceso obtiene la cpu, se ejecuta hasta terminar, ya que es una disciplina "no apropiativa".

Puede ocasionar que procesos largos hagan esperar a procesos cortos y que procesos no importantes hagan esperar a procesos importantes.

Es más predecible que otros esquemas.

No puede garantizar buenos tiempos de respuesta interactivos.

Suele utilizarse integrado a otros esquemas, por ejemplo, de la siguiente manera:

- Los procesos se despachan con algún esquema de prioridad.
- Los procesos con igual prioridad se despachan "FIFO".

2.12.4 Planificación de Asignación en Rueda (RR: Round Robin)

Los procesos se despachan en "FIFO" y disponen de una cantidad limitada de tiempo de cpu, llamada "división de tiempo" o "cuanto".

Si un proceso no termina antes de expirar su tiempo de cpu ocurren las siguientes acciones:

1. La cpu es apropiada.
2. La cpu es otorgada al siguiente proceso en espera.
3. El proceso apropiado es situado al final de la lista de listos.

Es efectiva en ambientes de tiempo compartido.

La sobrecarga de la apropiación se mantiene baja mediante mecanismos eficientes de intercambio de contexto y con suficiente memoria principal para los procesos.

2.12.5 Tamaño del Cuanto o Quantum

La determinación del tamaño del cuanto es decisiva para la operación efectiva de un sistema computacional [7, Deitel].

Los interrogantes son: ¿cuanto pequeño o grande?, ¿cuanto fijo o variable? y ¿cuanto igual para todos los procesos de usuarios o determinado por separado para cada uno de ellos?.

Si el cuanto se hace muy grande, cada proceso recibe todo el tiempo necesario para llegar a su terminación, por lo cual la asignación en rueda ("RR") degenera en "FIFO".

Si el cuanto se hace muy pequeño, la sobrecarga del intercambio de contexto se convierte en un factor dominante y el rendimiento del sistema se degrada, puesto que la mayor parte del tiempo de cpu se invierte en el intercambio del procesador (cambio de contexto) y los procesos de usuario disponen de muy poco tiempo de cpu.

El cuanto debe ser lo suficientemente grande como para permitir que la gran mayoría de las peticiones interactivas requieran de menos tiempo que la duración del cuanto, es decir que el tiempo transcurrido desde el otorgamiento de la cpu a un proceso hasta que genera una petición de Entrada / Salida debe ser menor que el cuanto establecido, de esta forma, ocurrida la petición la cpu pasa a otro proceso y como el cuanto es mayor que el tiempo transcurrido hasta la petición de Entrada / Salida, los procesos trabajan al máximo de velocidad, se minimiza la sobrecarga de apropiación y se maximiza la utilización de la Entrada / Salida.

El cuanto óptimo varía de un sistema a otro y con la carga, siendo un valor de referencia 100 mseg (cien milisegundos).

2.12.6 Planificación del Trabajo Más Corto Primero (SJF)

Es una disciplina no apropiativa y por lo tanto no recomendable en ambientes de tiempo compartido.

El proceso en espera con el menor tiempo estimado de ejecución hasta su terminación es el siguiente en ejecutarse.

Los tiempos promedio de espera son menores que con "FIFO".

Los tiempos de espera son menos predecibles que en "FIFO".

Favorece a los procesos cortos en detrimento de los largos.

Tiende a reducir el número de procesos en espera y el número de procesos que esperan detrás de procesos largos.

Requiere un conocimiento preciso del tiempo de ejecución de un proceso, lo que generalmente se desconoce.

Se pueden estimar los tiempos en base a series de valores anteriores.

2.12.7 Planificación del Tiempo Restante Más Corto (SRT)

Es la contraparte apropiativa del SJF.

Es útil en sistemas de tiempo compartido.

El proceso con el tiempo estimado de ejecución menor para finalizar es el siguiente en ser ejecutado.

Un proceso en ejecución puede ser apropiado por un nuevo proceso con un tiempo estimado de ejecución menor.

Tiene mayor sobrecarga que la planificación SJF.

Debe mantener un registro del tiempo de servicio transcurrido del proceso en ejecución, lo que aumenta la sobrecarga.

Los trabajos largos tienen un promedio y una varianza de los tiempos de espera aún mayor que en SJF.

La apropiación de un proceso a punto de terminar por otro de menor duración recién llegado podría significar un mayor tiempo de cambio de contexto (administración del procesador) que el tiempo de finalización del primero.

Al diseñarse los Sistemas Operativos se debe considerar cuidadosamente la sobrecarga de los mecanismos de administración de recursos comparándola con los beneficios esperados.

2.12.8 Planificación el Siguiente con Relación de Respuesta Máxima (HRN)

Corrige algunas de las debilidades del SJF, tales como el exceso de perjuicio hacia los procesos (trabajos) largos y el exceso de favoritismo hacia los nuevos trabajos cortos.

Es una disciplina no apropiativa.

La prioridad de cada proceso está en función no sólo del tiempo de servicio del trabajo, sino que también influye la cantidad de tiempo que el trabajo ha estado esperando ser servido.

Cuando un proceso ha obtenido la cpu, corre hasta terminar.

Las prioridades, que son dinámicas, se calculan según la siguiente fórmula, donde p_r es la "prioridad", t_e es el "tiempo de espera" y t_s es el "tiempo de servicio":

$$\boxed{p_r = \frac{(t_e + t_s)}{t_s}} \tag{2.1}$$

2.12.9 Planificación por Prioridad

Considera factores externos al proceso [23, Tanenbaum].

Las ideas centrales son que cada proceso tiene asociada una prioridad y que el proceso ejecutable con máxima prioridad es el que tiene el permiso de ejecución.

Los procesos de alta prioridad podrían ejecutar indefinidamente, ya que el planificador del sistema puede disminuir la prioridad del proceso en ejecución en cada interrupción del reloj.

Las prioridades también pueden ser asignadas dinámicamente por el sistema para lograr ciertas metas relacionadas con el procesador o la Entrada / Salida.

Los procesos limitados por la Entrada / Salida (requerimientos intensivos de Entrada / Salida) ocupan mucho de su tiempo en espera de operaciones de Entrada / Salida, por lo tanto:

- Deben tener prioridad para usar la cpu y efectuar la siguiente petición de Entrada / Salida, ya que se ejecutará (la operación de Entrada / Salida) en paralelo con otro proceso que utilice la cpu.
- Si deben esperar mucho tiempo a la cpu estarán ocupando memoria por un tiempo innecesario.

Un algoritmo sencillo consiste en establecer que la prioridad sea "$1/f$", donde "f" es la fracción del último cuanto utilizado por el proceso.

Un proceso que utilice 2 mseg (dos milisegundos) de su cuanto de 100 mseg (cien milisegundos) tendrá prioridad 50 (cincuenta).

Un proceso que se ejecutó 50 mseg antes del bloqueo tendrá prioridad 2.

Un proceso que utilizó todo el cuanto tendrá prioridad 1.

Frecuentemente los procesos se agrupan en "*Clases de Prioridad*", en cuyo caso se utiliza la Planificación con Prioridades entre las clases y con Round Robin (RR) dentro de cada clase. Si las prioridades no se reajustan en algún momento, los procesos de las clases de prioridad mínima podrían demorarse indefinidamente.

2.12.10 Colas de Retroalimentación de Niveles Múltiples

Proporcionan una estructura para lograr los siguientes objetivos:

- Favorecer trabajos cortos.
- Favorecer trabajos limitados por la Entrada / Salida para optimizar el uso de los dispositivos de Entrada / Salida.
- Determinar la naturaleza de un trabajo lo más rápido posible y planificar el trabajo (proceso) en consecuencia.

Un nuevo proceso entra en la red de línea de espera al final de la cola superior.

Se mueve por esta cola "FIFO" hasta obtener la cpu.

Si el trabajo termina o abandona la cpu para esperar por la terminación de una operación de Entrada / Salida o la terminación de algún otro suceso, el trabajo abandona la red de línea de espera.

Si su cuanto expira antes de abandonar la cpu voluntariamente, el proceso se coloca en la parte trasera de la cola del siguiente nivel inferior.

El trabajo recibe servicio al llegar a la cabeza de esta cola si la primera está vacía.

Mientras el proceso continúe consumiendo totalmente su cuanto en cada nivel, continuará moviéndose hacia el final de las colas inferiores.

Generalmente hay una cola en la parte más profunda a través de la cual el proceso circula en asignación de rueda hasta que termina.

Existen esquemas en los que el cuanto otorgado al proceso aumenta a medida que el proceso se mueve hacia las colas de los niveles inferiores, en tal caso, cuanto más tiempo haya estado el proceso en la red de línea de espera, mayor será su cuanto cada vez que obtiene la cpu y no podrá obtener la cpu muy a menudo debido a la mayor prioridad de los procesos de las colas superiores.

Un proceso situado en una cola dada no podrá ser ejecutado hasta que las colas de los niveles superiores estén vacías.

Un proceso en ejecución es apropiado por un proceso que llegue a una cola superior.

Es un mecanismo adaptable, es decir que se adapta a cargas variables.

A los efectos de una revisión gráfica de lo enunciado precedentemente, ver la figura 2.5 de la página 47 [7, Deitel].

2.12.11 Política Versus Mecanismo de Planificación

Puede ocurrir que haya procesos con muchos procesos hijos ejecutándose bajo su control, por ejemplo, un proceso en un DBMS con procesos hijos atendiendo funciones específicas, tales como, análisis de interrogantes, acceso a discos, etc.

Es posible que el proceso principal (padre) pueda identificar la importancia (o criticidad) de sus procesos hijos, pero los planificadores analizados no aceptan datos de los procesos de usuario relativos a decisiones de planificación.

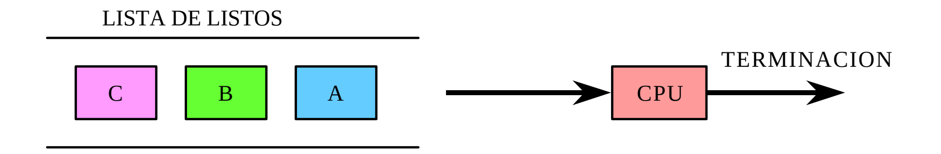
La solución es separar el **mecanismo de planificación** de la **política de planificación** , para ello se parametriza el algoritmo de planificación y los parámetros pueden ser determinados por medio de procesos del usuario; así el mecanismo está en el núcleo del Sistema Operativo pero la política queda establecida por un proceso del usuario.

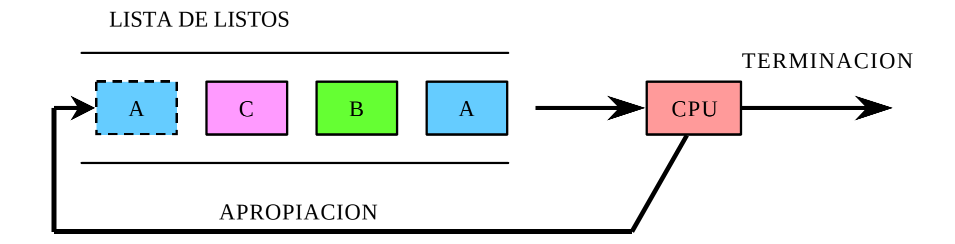
2.12.12 Planificación de Dos Niveles

Los esquemas analizados hasta ahora suponen que todos los procesos ejecutables están en la memoria principal.

Si la memoria principal es insuficiente, ocurrirá lo siguiente [23, Tanenbaum]:

- Habrá procesos ejecutables que se mantengan en disco.
- Habrá importantes implicaciones para la planificación, tales como las siguientes:
  - El tiempo de alternancia entre procesos para traer y procesar un proceso del disco es considerablemente mayor que el tiempo para un proceso que ya está en la memoria principal.
  - Es más eficiente el intercambio de los procesos con un planificador de dos niveles.





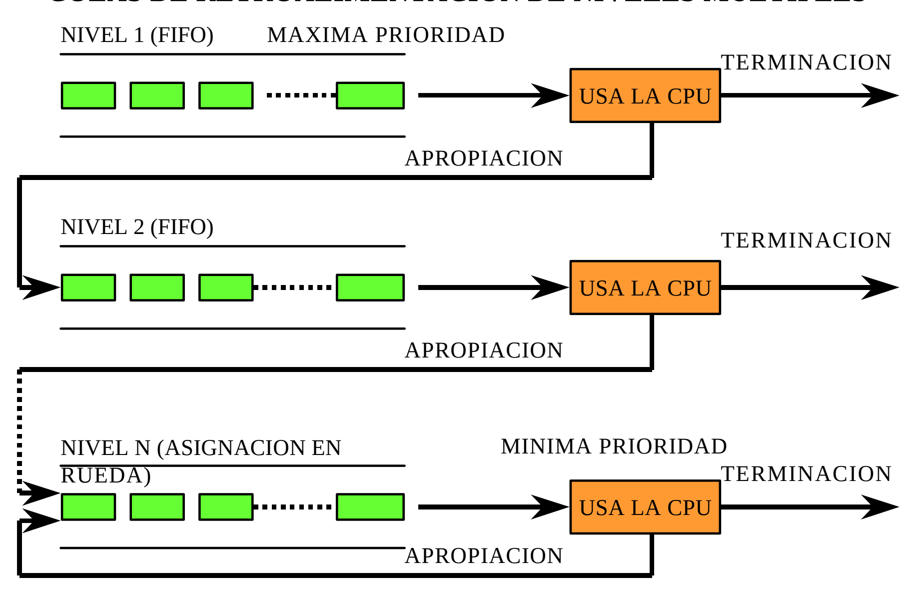


Figura 2.5: Tipos de planificación del procesador.

El esquema operativo de un planificador de dos niveles es como sigue:

1. Se carga en la memoria principal cierto subconjunto de los procesos ejecutables.

2. El planificador se restringe a ellos durante cierto tiempo.

3. Periódicamente se llama a un planificador de nivel superior para efectuar las siguientes tareas:

   (a) Eliminar de la memoria los procesos que hayan permanecido en ella el tiempo suficiente.

   (b) Cargar a memoria los procesos que hayan estado en disco demasiado tiempo.

4. El planificador de nivel inferior se restringe de nuevo a los procesos ejecutables que se encuentren en la memoria.

5. El planificador de nivel superior se encarga de desplazar los procesos de memoria a disco y viceversa.

Los criterios que podría utilizar el planificador de nivel superior para tomar sus decisiones son los que se indican a continuación:

- ¿Cuánto tiempo ha transcurrido desde el último intercambio del proceso?.

- ¿Cuánto tiempo de cpu ha utilizado recientemente el proceso?.

- ¿Qué tan grande es el proceso? (generalmente los procesos pequeños no causan tantos problemas en este sentido).

- ¿Qué tan alta es la prioridad del proceso?.

El planificador de nivel superior podría utilizar cualquiera de los métodos de planificación analizados.

2.13 Multiprocesamiento

2.13.1 Introducción

Es una tendencia significativa en el campo de la computación.

Consiste en configurar un sistema de computación con varios procesadores .

No es un enfoque nuevo pero sí posee grandes perspectivas en función del desarrollo de los microprocesadores.

Se podrían concebir sistemas construidos por cientos o miles de microprocesadores.

2.13.2 Confiabilidad

Si un procesador falla, los restantes continúan operando, lo cual no es automático y requiere de un diseño cuidadoso.

Un procesador que falla habrá de informarlo a los demás de alguna manera, para que se hagan cargo de su trabajo .

Los procesadores en funcionamiento deben poder detectar el fallo de un procesador determinado.

El Sistema Operativo debe percibir que ha fallado un procesador determinado y ya no podrá asignarlo y también debe ajustar sus estrategias de asignación de recursos para evitar la sobrecarga del sistema que está degradado.

2.13.3 Explotación del Paralelismo

La mayoría de los sistemas de multiprocesamiento tienen como meta principal el incremento de la capacidad de ejecución.

La programación sigue siendo esencialmente secuencial y generalmente no se explota la concurrencia.

Las principales razones son las siguientes:

- Las personas piensan en forma secuencial.
- Ningún lenguaje humano proporciona la expresión adecuada de paralelismo, pero existen lenguajes de computación con soporte de concurrencia (por ejemplo, Ada, Pascal Concurrente, etc.).
- Ni el multiprocesamiento ha sido usado con amplitud para explotar el paralelismo.
- El hardware tradicional del computador está orientado hacia la operación secuencial.
- Es muy difícil depurar programas en paralelo.

Los multiprocesadores no se utilizan a menudo para explotar el paralelismo ya que es muy escaso el software que explote el paralelismo.

Lo deseable es que los Sistemas Operativos y compiladores puedan detectar e implementar el paralelismo automáticamente.

2.13.4 Paralelismo Masivo

Se debe disponer de suficientes procesadores como para que todas las operaciones que puedan ser ejecutadas en paralelo puedan ser asignadas a procesadores separados [14, Pino y Marrone].

Esto ofrece una forma de ejecutar un programa en el menor tiempo posible.

La cuestión central es, disponiendo del paralelismo masivo, ¿cuál es el tiempo mínimo requerido para ejecutar un algoritmo determinado?.

2.13.5 Metas de los Sistemas de Multiprocesamiento

Las metas de los sistemas de multiprocesamiento generalmente son la confiabilidad y la disponibilidad muy altas, como así también el incremento del poder de computación.

El diseño modular proporciona una flexibilidad importante y facilita la expansión de la capacidad.

2.13.6 Detección Automática del Paralelismo

Los multiprocesadores hacen posible la explotación del paralelismo.

Los sistemas de computación obtienen los beneficios del procesamiento concurrente más por la "*multiprogramación*" de varios procesos y menos por la explotación del "*paralelismo*" dentro de un solo proceso.

La detección del paralelismo es un problema complejo y la puede efectuar el programador, el traductor del lenguaje, el hardware o el Sistema Operativo.

El paralelismo dentro de los programas puede ser "*explícito*" o "*implícito*".

Las principales características del *paralelismo explícito* son las que se detallan a continuación:

- Es indicado de forma específica por un programador mediante una "construcción de concurrencia" como la siguiente:

```
cobegin;
        proposición 1;
        ................
        proposición n;
coend;
```

- Se pueden utilizar procesadores separados para ejecutar cada una de las proposiciones.

- Es susceptible de errores de programación difíciles de detectar y depurar.

- El programador puede omitir tratar situaciones donde sería aplicable el paralelismo.

Las principales características del *paralelismo implícito* son las que se detallan a continuación:

- La verdadera esperanza está en la detección automática del paralelismo implícito.

- Es el paralelismo intrínseco del algoritmo pero no establecido explícitamente por el programador.

- Los compiladores explotan el paralelismo implícito mediante las técnicas de "distribución de ciclos" y de "reducción de la altura del árbol".

2.13.7 Distribución de Ciclos

Una "estructura de ciclos o de repetición" implica la repetición de una serie de proposiciones (cuerpo del ciclo) hasta que ocurre alguna condición de terminación, por ejemplo:

For $i = 1$ to 3

Do

$$\boxed{a(i) = b(i) + c(i)} \tag{2.2}$$

;

El procesador secuencial realizará en secuencia lo siguiente:

$$\begin{aligned} a(1) &= b(1) + c(1) \\ a(2) &= b(2) + c(2) \\ a(3) &= b(3) + c(3) \end{aligned}$$

En un sistema de multiprocesamiento con tres procesadores disponibles se podrían ejecutar concurrentemente.

Un compilador que detecte automáticamente el paralelismo implícito puede convertir el ciclo del ejemplo 2.2 de la página 51 en lo siguiente:

cobegin;

$$\begin{aligned} a(1) &= b(1) + c(1) \\ a(2) &= b(2) + c(2) \\ a(3) &= b(3) + c(3) \end{aligned}$$

coend;

Esta técnica se denomina **distribución de ciclos**.

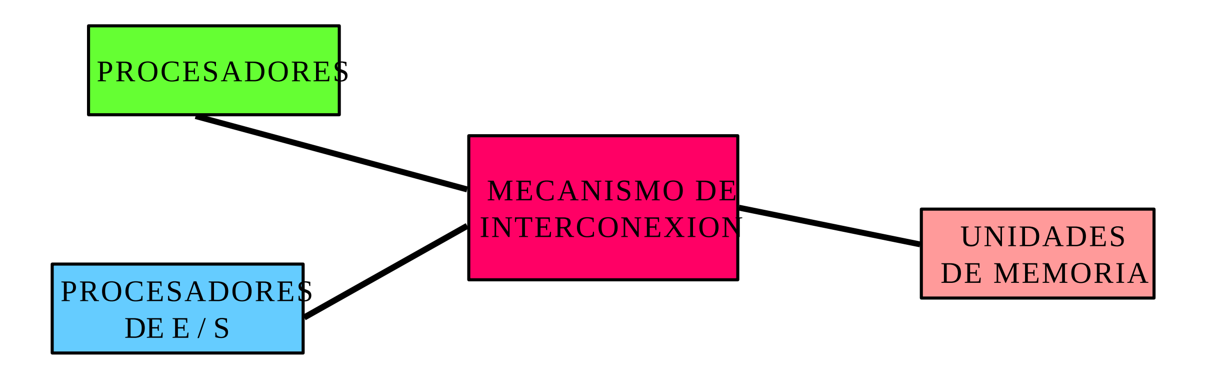


Figura 2.6: Idea simplificada de la organización de un multiprocesador.

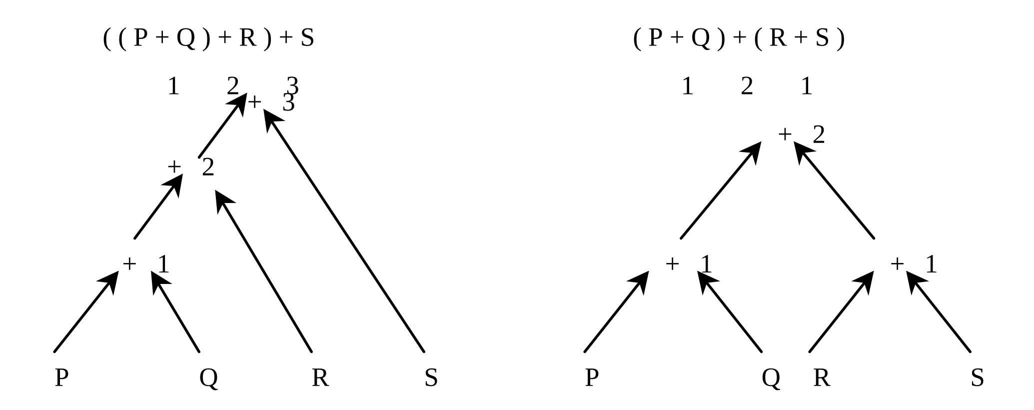


Figura 2.7: Reducción de la altura del árbol por asociatividad.

2.13.8 Reducción de la Altura del Arbol

Utilizando las propiedades asociativa, conmutativa y distributiva de la aritmética, los compiladores pueden:

1. Detectar el paralelismo implícito en expresiones algebraicas.
2. Producir un código objeto para multiprocesadores que indique las operaciones que se pueden realizar simultáneamente.
3. Reordenar expresiones para que sean más apropiadas para la computación en paralelo.

Se invierten más tiempo y recursos durante la compilación para reducir el tiempo de ejecución, es decir que se busca optimización en el momento de la compilación para lograr ejecución en tiempo mínimo, lo que es aplicable especialmente cuando los sistemas pasan a producción, no tanto cuando están en desarrollo[9].

[9] Ver Figura 2.6 de la página 52, Figura 2.7 de la página 52, Figura 2.8 de la página 53 y Figura 2.9 de la página 53 [7, Deitel].

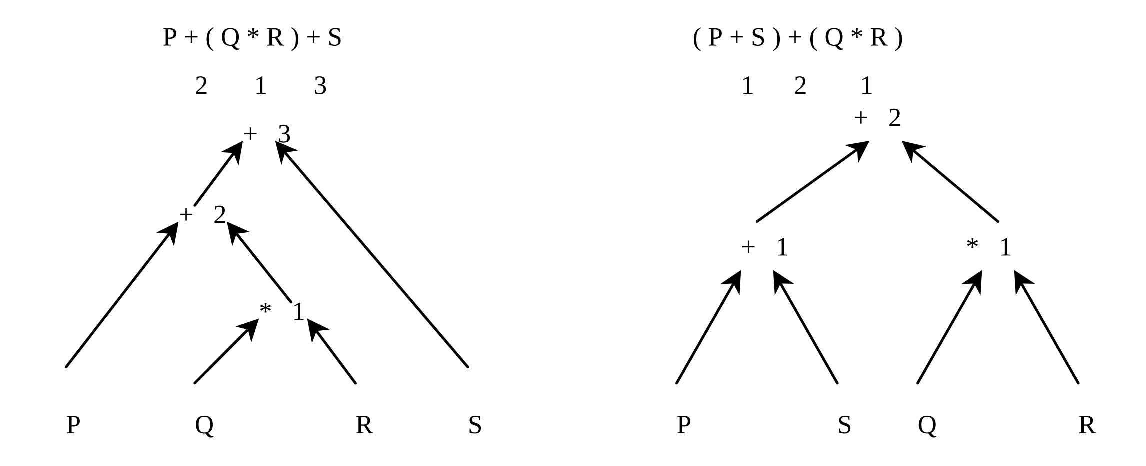


Figura 2.8: Reducción de la altura del árbol por conmutatividad.

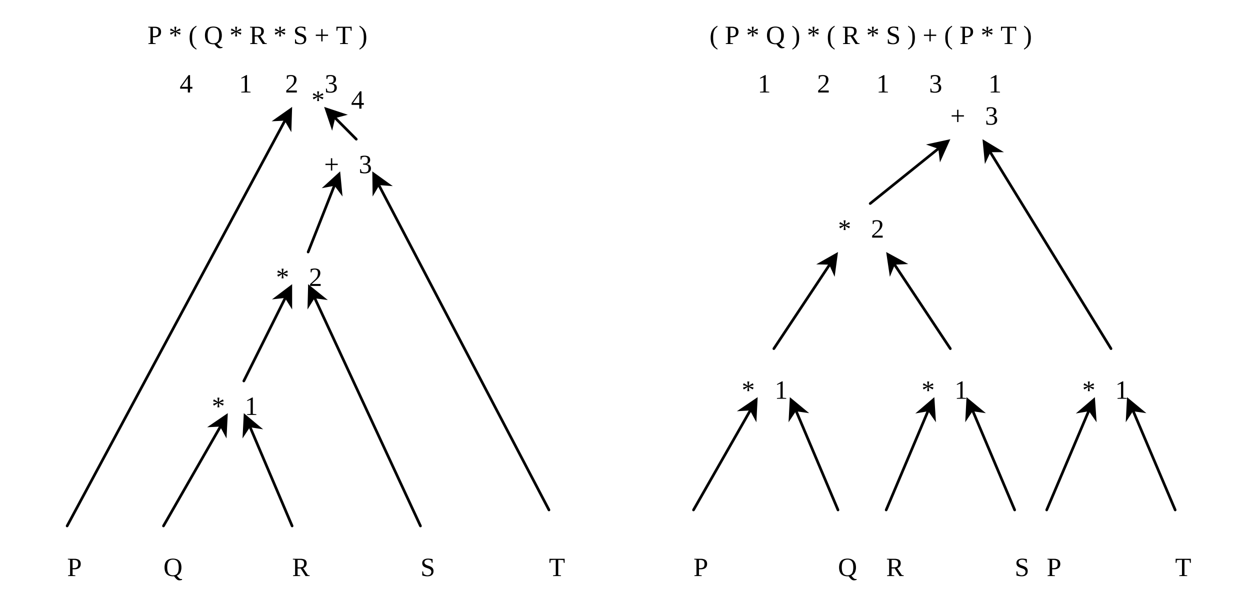


Figura 2.9: Reducción de la altura del árbol por distributividad.

REGLA DE "NUNCA ESPERAR": Es mejor darle a un procesador una tarea que puede llegar a no ser utilizada, que tenerlo ocioso.

2.14 Organización del Hardware del Multiprocesador

El problema clave es determinar los medios de conexión de los procesadores múltiples y los procesadores de Entrada / Salida a las unidades de almacenamiento [7, Deitel].

Los multiprocesadores se caracterizan por los siguientes aspectos:

- Un multiprocesador contiene dos o más procesadores con capacidades aproximadamente comparables.
- Todos los procesadores comparten el acceso a un almacenamiento común y a canales de Entrada / Salida, unidades de control y dispositivos.
- Todo está controlado por un Sistema Operativo que proporciona interacción entre procesadores y sus programas en los niveles de trabajo, tarea, paso, archivo y elementos de datos.

Las organizaciones más comunes son las siguientes:

- Tiempo compartido o bus común (conductor común).
- Matriz de barras cruzadas e interruptores.
- Almacenamiento de interconexión múltiple.

2.14.1 Tiempo Compartido o Bus Común (o Conductor Común)

Usa un solo camino de comunicación entre todas las unidades funcionales[10].

El bus común es en esencia una unidad pasiva.

Un procesador o procesador de Entrada / Salida que desee transferir datos debe efectuar los siguientes pasos:

1. Verificar la disponibilidad del conductor y de la unidad de destino.
2. Informar a la unidad de destino de lo que se va a hacer con los datos.
3. Iniciar la transferencia de datos.

Las unidades receptoras deben poder reconocer qué mensajes del bus son enviados hacia ellas y seguir y confirmar las señales de control recibidas de la unidad emisora.

Es una organización económica, simple y flexible pero con una sola vía de comunicación, por lo cual:

- El sistema falla totalmente si falla el bus.

[10] Ver Figura 2.10 de la página 55 [7, Deitel].

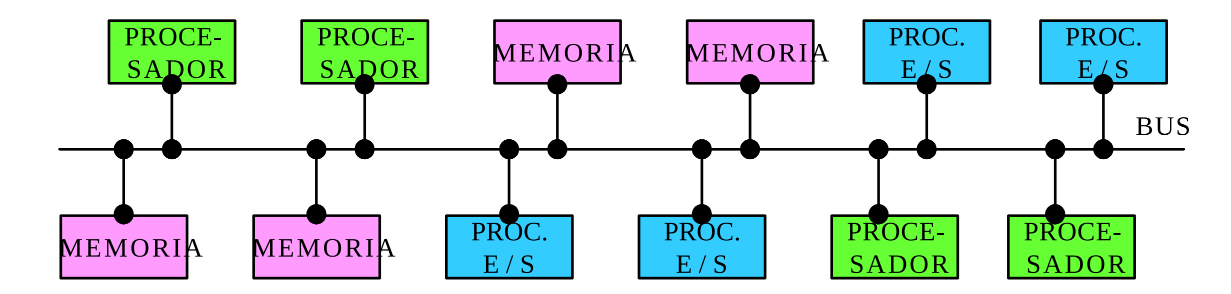


Figura 2.10: Organización de multiprocesador de tiempo compartido de bus común.

- La tasa neta de transmisiones está limitada por la tasa neta de transmisión del conductor.
- La contención por el uso del bus en un sistema sobrecargado puede ocasionar una seria degradación.

2.14.2 Matriz de Barras Cruzadas e Interruptores

En este caso existe un camino diferente para cada unidad de almacenamiento, por lo cual las referencias a dos unidades diferentes de almacenamiento no son bloqueantes sino simultáneas y la multiplicidad de caminos de transmisión puede proporcionar tasas de transferencia muy altas[11].

2.14.3 Almacenamiento de Interconexión Múltiple

Se obtiene al sacar las lógicas de control, de conmutación y de arbitraje de prioridades fuera del interruptor de barras cruzadas y se las coloca en la interfaz de cada unidad de almacenamiento[12].

Cada unidad funcional puede acceder a cada unidad de almacenamiento, pero sólo en una "*conexión de almacenamiento*" específica, es decir que hay una conexión de almacenamiento por unidad funcional.

El conexionado es más complejo que en los otros esquemas.

Se puede restringir el acceso a las unidades de almacenamiento para que no todas las unidades de procesamiento las accedan, en tal caso habrá unidades de almacenamiento "privadas" de determinados procesadores[13].

2.15 Grados de Acoplamiento en Multiprocesamiento

Los grados de acoplamiento en multiprocesamiento pueden clasificarse de **ligeramente acoplados**[14] o **rígidamente acoplados** [15], según las características que se detallan en

[11] Ver Figura 2.11 de la página 56 [7, Deitel].
[12] Ver Figura 2.12 de la página 56 [7, Deitel].
[13] Ver Figura 2.13 de la página 57 [7, Deitel].
[14] Ver Figura 2.14 de la página 58 [7, Deitel].
[15] Ver Figura 2.15 de la página 59 [7, Deitel].

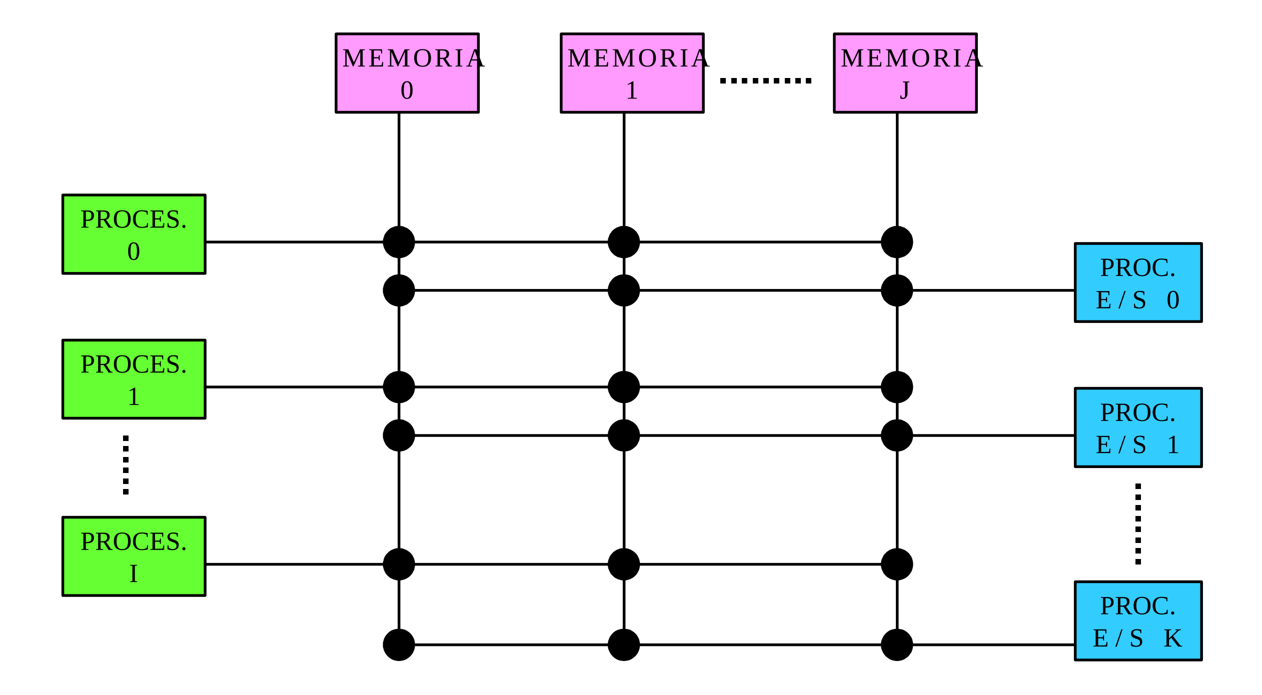


Figura 2.11: Organización del multiprocesador por matriz de barras cruzadas e interruptores.

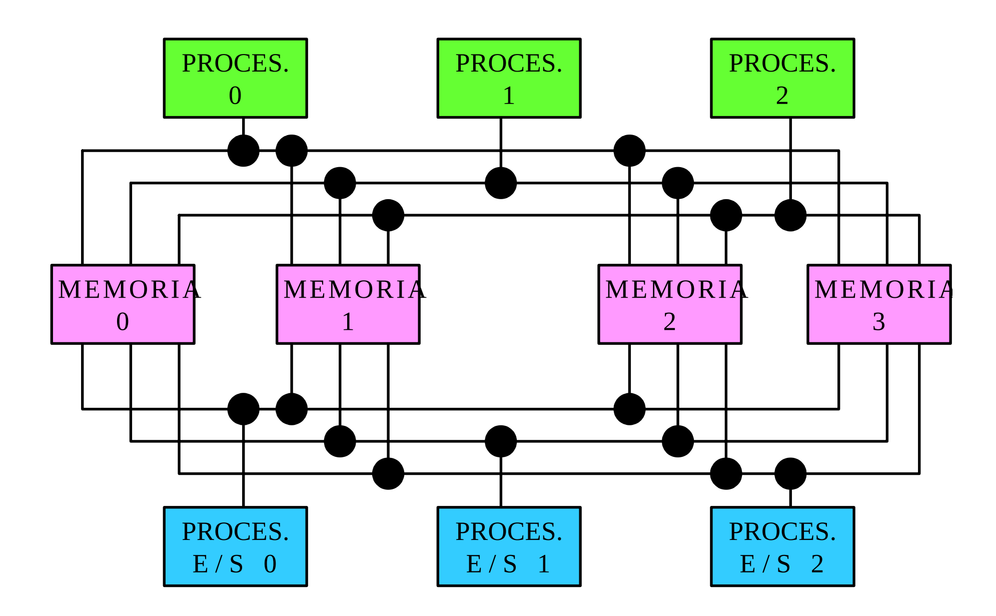


Figura 2.12: Organización de multiprocesador por sistema de memoria de interconexión múltiple.

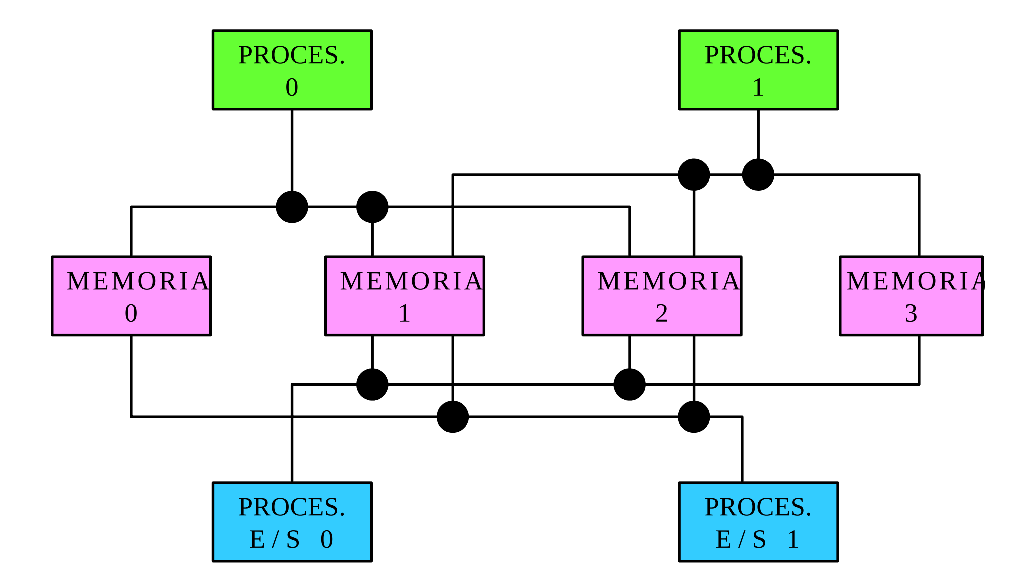


Figura 2.13: Organización de multiprocesador por sistema de memoria de interconexión múltiple con memorias privadas.

la tabla 2.5 de la página 58 [7, Deitel].

2.15.1 Organización Maestro / Satélite

Un procesador está diseñado como el "maestro" y los otros como "satélites".

El procesador "maestro" es de propósito general y realiza operaciones de Entrada / Salida y computaciones.

Los procesadores "satélites" sólo realizan computaciones.

Los procesos limitados por computación pueden ejecutarse con efectividad en los satélites.

Los procesos limitados por la Entrada / Salida ejecutados en los satélites generan frecuentes llamadas de servicios al procesador maestro, pudiendo resultar ineficientes.

Si falla un satélite se pierde capacidad computacional pero el sistema no falla.

Si falla el maestro el sistema falla al no poder efectuar operaciones de Entrada / Salida, por lo que un satélite debería asumir las funciones del maestro previo cambio de los periféricos y reinicio del sistema.

En el multiprocesamiento simétrico todos pueden hacer Entrada / Salida.

2.16 Sistema Operativo de Multiprocesadores

Las capacidades funcionales de los Sistema Operativo de multiprogramación y de multiprocesadores incluyen lo siguiente:

- Asignación y administración de recursos.
- Protección de tablas y conjuntos de datos.

Grados de acoplamiento en multiprocesamiento	Descripción
Ligeramente acoplado	Incluye la conexión de dos o más sistemas independientes por medio de un enlace de comunicación. Cada sistema tiene su propio Sistema Operativo y almacenamiento. Los sistemas pueden funcionar independientemente y se comunican cuando sea necesario. Los sistemas separados pueden acceder a los archivos de los otros e intercambiar tareas a procesadores menos cargados.
Rígidamente acoplado	Utiliza un solo almacenamiento compartido por varios procesadores. Emplea un solo Sistema Operativo que controla todos los procesadores y el hardware del sistema.

Tabla 2.5: Grados de acoplamiento en multiprocesamiento.

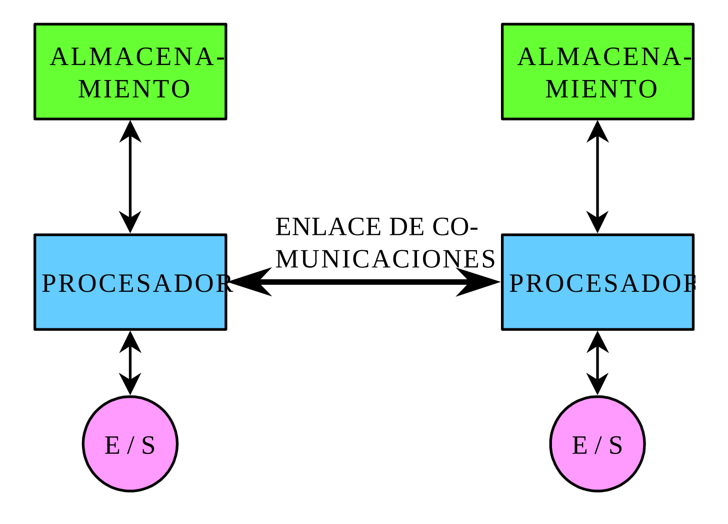


Figura 2.14: Multiprocesamiento ligeramente acoplado.

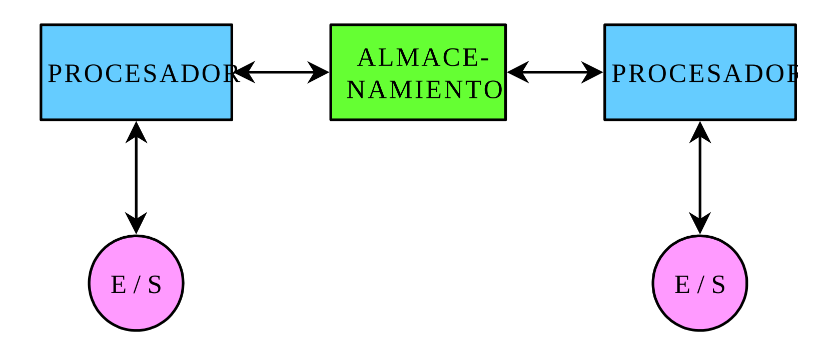


Figura 2.15: Multiprocesamiento rígidamente acoplado.

- Prevención contra el interbloqueo del sistema.
- Terminación anormal.
- Equilibrio de cargas de Entrada / Salida.
- Equilibrio de carga del procesador.
- Reconfiguración.

Las tres últimas son especialmente importantes en Sistemas Operativos de multiprocesadores, donde es fundamental explotar el paralelismo en el hardware y en los programas y hacerlo automáticamente.

Las organizaciones básicas de los Sistemas Operativos para multiprocesadores son las siguientes:

- Maestro / satélite.
- Ejecutivo separado para cada procesador.
- Tratamiento simétrico (o anónimo) para todos los procesadores.

2.16.1 Maestro / Satélite

Es la organización más fácil de implementar.

No logra la utilización óptima del hardware dado que sólo el procesador maestro puede ejecutar el Sistema Operativo y el procesador satélite sólo puede ejecutar programas del usuario.

Las interrupciones generadas por los procesos en ejecución en los procesadores satélites que precisan atención del Sistema Operativo deben ser atendidas por el procesador maestro y por ello pueden generarse largas colas de requerimientos pendientes.

2.16.2 Ejecutivos Separados

Cada procesador tiene su propio Sistema Operativo y responde a interrupciones de los usuarios que operan en ese procesador.

Existen tablas de control con información global de todo el sistema (por ejemplo, lista de procesadores conocidos por el Sistema Operativo) a las que se debe acceder utilizando exclusión mutua.

Es más confiable que la organización maestro / satélite.

Cada procesador controla sus propios recursos dedicados.

La reconfiguración de los dispositivos de Entrada / Salida puede implicar el cambio de dispositivos a diferentes procesadores con distintos Sistemas Operativos.

La contención sobre las tablas del Sistema Operativo es mínima.

Los procesadores no cooperan en la ejecución de un proceso individual, que habrá sido asignado a uno de ellos.

2.16.3 Tratamiento Simétrico

Es la organización más complicada de implementar y también la más poderosa y confiable.

El Sistema Operativo administra un grupo de procesadores idénticos, donde cualquiera puede utilizar cualquier dispositivo de Entrada / Salida y cualquiera puede referenciar a cualquier unidad de almacenamiento.

El Sistema Operativo precisa código reentrante y exclusión mutua.

Es posible equilibrar la carga de trabajo más precisamente que en las otras organizaciones.

Adquieren significativa importancia el hardware y el software para resolución de conflictos.

Todos los procesadores pueden cooperar en la ejecución de un proceso determinado.

El **procesador ejecutivo** es el responsable (uno sólo) en un momento dado de las tablas y funciones del sistema; así se evitan los conflictos sobre la información global.

2.17 Rendimiento del Sistema de Multiprocesamiento

Aún con multiprocesamiento completamente simétrico, la adición de un nuevo procesador no hará que la capacidad de ejecución del sistema aumente según la capacidad del nuevo procesador, siendo las causas las siguientes:

- Hay sobrecarga adicional del Sistema Operativo.
- Se incrementa la contención por recursos del sistema.
- Hay retrasos del hardware en el intercambio y en el encaminamiento de las transmisiones entre un número mayor de componentes.

Al incrementar el número de procesadores “n” similares en un multiprocesador, el incremento de la productividad no es lineal y tiende a disminuir cuando “n” crece.

2.18 Recuperación de Errores

Una de las capacidades más importantes de los Sistemas Operativos de multiprocesadores es la de soportar fallas de hardware en procesadores individuales y continuar su operación.

Debe existir el soporte correspondiente en el Sistema Operativo.

Las técnicas de recuperación de errores incluyen los siguientes aspectos:

- Los datos críticos (del sistema y de usuario) deben mantenerse en copias múltiples y en bancos de almacenamiento separados.
- El Sistema Operativo debe ejecutar efectivamente con la configuración máxima y con subconjuntos ante fallas.
- Debe haber capacidad de detección y corrección de errores de hardware sin interferir con la eficiencia operacional del sistema.
- Se debe utilizar la capacidad ociosa del procesador para tratar de detectar posibles fallos antes de que se produzcan.
- El Sistema Operativo debe dirigir un procesador operativo para que tome el control de un proceso que se estaba ejecutando en un procesador que falla.

2.19 Multiprocesamiento Simétrico (MPS)

Cada procesador posee capacidades funcionales completas.

Los dispositivos de Entrada / Salida pueden ser conectados a cada uno de los procesadores[16].

Todas las llamadas al supervisor pueden ser ejecutadas en todos los procesadores, inclusive las de Entrada / Salida.

Si un programa en ejecución en un procesador pide una operación de Entrada / Salida en un dispositivo conectado a un procesador diferente, el procesador puede continuar ejecutando el trabajo y la Entrada / Salida se coloca en una cola para su iniciación por el procesador apropiado.

Se considera *procesador ejecutante* al que está ejecutando un proceso determinado.

Se considera *procesador propietario* al que está conectado a los diferentes dispositivos utilizados por el proceso.

Es más eficiente que la organización maestro / satélite, ya que los requerimientos de Entrada / Salida se encolan y no sobrecargan con intercambio de contexto y a que en la organización maestro / satélite las peticiones de Entrada / Salida en el satélite provocan un intercambio de contexto en el maestro.

Cada procesador puede ejecutar el planificador para buscar el siguiente trabajo a ejecutar, de forma que un proceso determinado se ejecuta en diferentes procesadores en distintos momentos; además, el MPS utiliza una sola cola de trabajos y cada procesador puede seleccionar trabajos de ella, con lo cual se equilibra la carga entre los procesadores.

Para minimizar la contención en el despacho de procesos, los relojes de los procesadores tienen oblicuidad, debido a ello las interrupciones de reloj ocurren en diferentes momentos.

2.20 Tendencias de los Multiprocesadores

Todo indica que el uso de los multiprocesadores se incrementará considerablemente en el futuro [7, Deitel].

[16] Ver Figura 2.16 de la página 62 [7, Deitel].

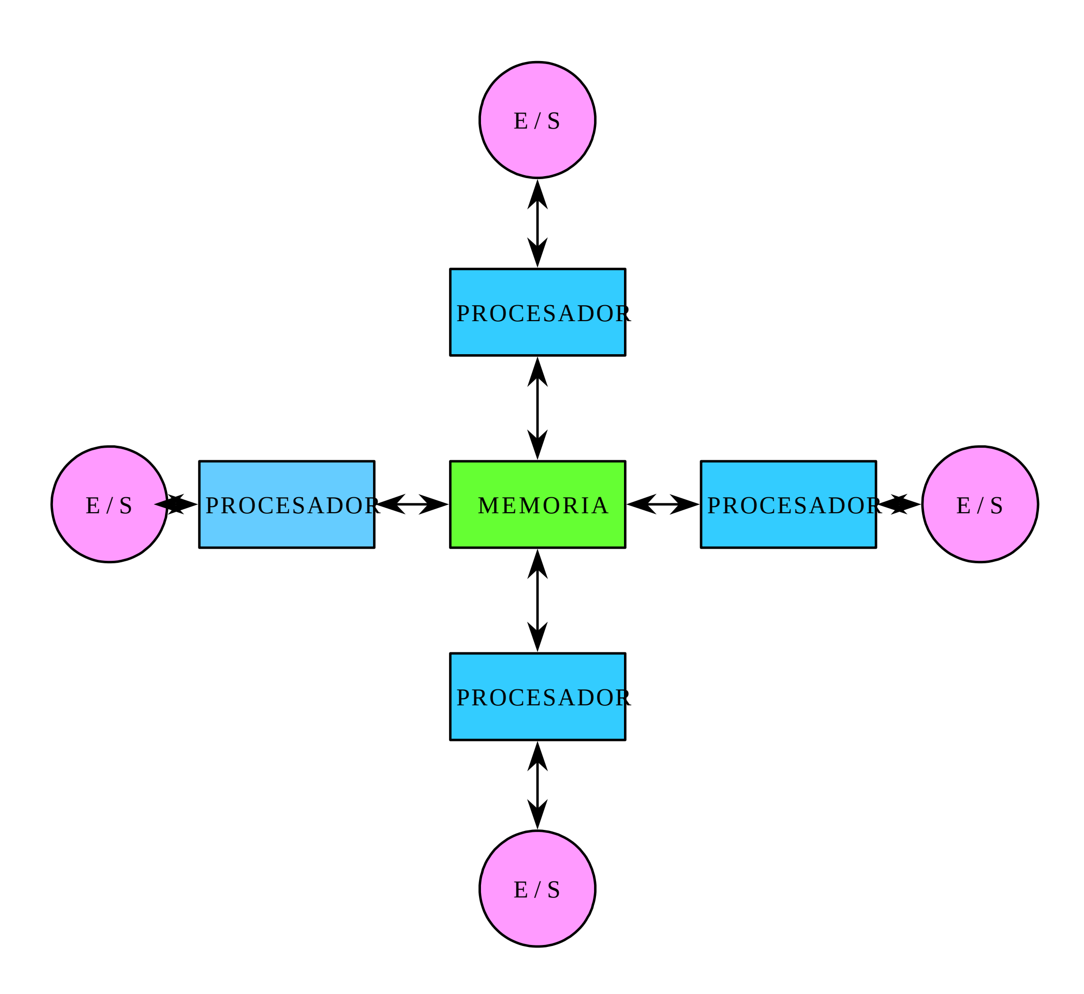


Figura 2.16: Ejemplo de implementación de multiprocesamiento simétrico.

Las principales razones son las siguientes:

- La confiabilidad requerida es cada vez mayor.
- La reducción de costos consecuencia de los avances en microelectrónica.
- El previsible desarrollo de lenguajes que permitan a los usuarios expresar el paralelismo explícitamente.
- El progreso en la detección automática del paralelismo .
- El hecho de que se estaría llegando a los límites del uniprocesador debido a la compactación de componentes, es decir que se estaría próximo a los límites de longitud y de proximidad de los "caminos electromagnéticos" (longitud del recorrido de la señal electromagnética); alcanzados los límites mencionados, la única posibilidad de incrementar capacidad de cómputo es mediante multiprocesamiento.

Existen estudios de tendencias en arquitectura de computadoras que apuntan a los poliprocesadores, es decir, sistemas que combinan el multiprocesamiento, simétrico y asimétrico, para crear una jerarquía de procesadores dentro de un sistema.

Capítulo 3

Administración de la Memoria

3.1 Introducción al Almacenamiento Real

La organización y administración de la "*memoria principal*", "*memoria primaria*" o "*memoria real*" de un sistema ha sido y es uno de los factores más importantes en el diseño de los S. O. [7, Deitel].

Los términos "*memoria*" y "*almacenamiento*" se consideran equivalentes.

Los programas y datos deben estar en el almacenamiento principal para:

- Poderlos ejecutar.
- Referenciarlos directamente.

Se considera "*almacenamiento secundario*" o "*almacenamiento auxiliar*" al generalmente soportado en discos.

Los hechos demuestran que generalmente los programas crecen en requerimientos de memoria tan rápido como las memorias:

- "Ley de Parkinson parafraseada": Los programas se desarrollan para ocupar toda la memoria disponible para ellos.

La parte del S. O. que administra la memoria se llama "*administrador de la memoria*":

- Lleva un registro de las partes de memoria que se están utilizando y de aquellas que no.
- Asigna espacio en memoria a los procesos cuando estos la necesitan.
- Libera espacio de memoria asignada a procesos que han terminado.

3.2 Organización y Administración del Almacenamiento

3.2.1 Organización del Almacenamiento

Históricamente el almacenamiento principal se ha considerado como un recurso costoso, por lo cual su utilización debía optimizarse [7, Deitel].

Por organización del almacenamiento se entiende la manera de considerar este almacenamiento:

- ¿ se coloca un solo programa de usuario o varios ?.
- Si se encuentran varios programas de usuario:
  - ¿ se concede a cada uno la misma cantidad de espacio o se divide el almacenamiento en porciones o "*particiones*" de diferente tamaño ?.
  - ¿ se utilizará un esquema rígido de número y tamaño de particiones o un esquema dinámico y adaptable ?.
  - ¿ se requerirá que los trabajos de los usuarios sean diseñados para funcionar en una partición específica o se permitirá que se ejecuten en cualquiera donde quepan ?.
  - ¿ se requerirá o no que cada trabajo sea colocado en un bloque contiguo de memoria ?.

3.2.2 Administración del Almacenamiento

Independientemente del esquema de organización hay que decidir las estrategias que se utilizarán para optimizar el rendimiento.

Las "*estrategias de administración*" deben considerar:

- ¿ cuándo se consigue un nuevo programa para colocar en la memoria ?:
  - ¿ cuando el sistema lo pide específicamente o se intenta anticiparse a las peticiones ?.
- ¿ dónde se colocará el programa que se ejecutará a continuación ?:
  - ¿ se prioriza el tiempo de carga o la optimización en el uso del almacenamiento ?.
- ¿ con qué criterio se desplazarán programas ?.

3.3 Jerarquía de Almacenamiento

Los programas y datos tienen que estar en la memoria principal para poder ejecutarse o ser referenciados [7, Deitel].

Los programas y datos que no son necesarios de inmediato pueden mantenerse en el almacenamiento secundario.

El almacenamiento principal es más costoso y menor que el secundario pero de acceso más rápido.

Los sistemas con varios niveles de almacenamiento requieren destinar recursos para administrar el movimiento de programas y datos entre niveles[1].

Un nivel adicional es el "*caché*" o **memoria de alta velocidad**, que posee las siguientes características:

[1] Ver Figura 3.1 de la página 67 [7, Deitel].

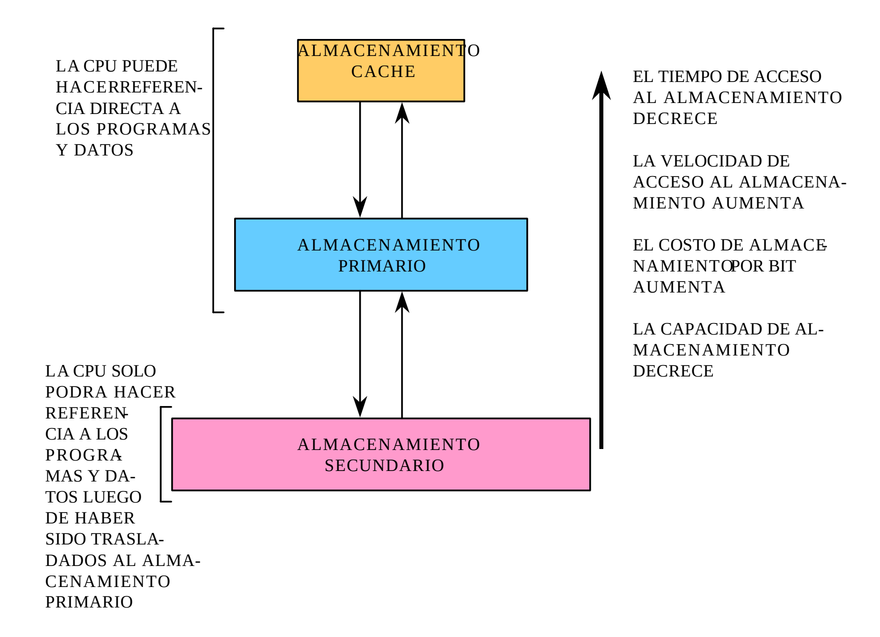


Figura 3.1: Organización jerárquica del almacenamiento.

- Es más rápida y costosa que la memoria principal.
- Impone al sistema un nivel más de traspaso:
  - Los programas son traspasados de la memoria principal al caché antes de su ejecución.
- Los programas en la memoria caché ejecutan mucho más rápido que en la memoria principal.
- Al utilizar memoria caché se espera que:
  - La sobrecarga que supone el traspaso de programas de un nivel de memoria a otro sea mucho menor que la mejora en el rendimiento obtenida por la posibilidad de una ejecución mucho más rápida en la caché.

3.4 Estrategias de Administración del Almacenamiento

Están dirigidas a la obtención del mejor uso posible del recurso del almacenamiento principal [7, Deitel].

Se dividen en las siguientes **categorías**:

- Estrategias de búsqueda:
  - Estrategias de búsqueda por demanda.

  - Estrategias de búsqueda anticipada.

- Estrategias de colocación.

- Estrategias de reposición.

Las "*estrategias de búsqueda*" están relacionadas con el hecho de cuándo obtener el siguiente fragmento de programa o de datos para su inserción en la memoria principal.

En la "*búsqueda por demanda*" el siguiente fragmento de programa o de datos se carga al almacenamiento principal cuando algún programa en ejecución lo referencia.

Se considera que la "*búsqueda anticipada*" puede producir un mejor rendimiento del sistema.

Las "*estrategias de colocación*" están relacionadas con la determinación del lugar de la memoria donde se colocará (cargará) un programa nuevo.

Las "*estrategias de reposición*" están relacionadas con la determinación de qué fragmento de programa o de datos desplazar para dar lugar a los programas nuevos.

3.4.1 Asignación Contigua de Almacenamiento Versus No Contigua

En la "*asignación contigua*" cada programa ocupa un bloque contiguo y sencillo de localizaciones de almacenamiento.

En la "*asignación no contigua*" un programa se divide en varios bloques o "*segmentos*" que pueden almacenarse en direcciones que no tienen que ser necesariamente adyacentes, por lo que es más compleja pero más eficiente que la asignación continua.

3.4.2 Asignación Contigua de Almacenamiento de Un Solo Usuario

Se consideran S. O. que ya poseen desarrollado el "*sistema de control de entrada / salida*": IOCS: input / output control system[2].

El tamaño de los programas está limitado por la cantidad de memoria principal, pero se puede superar este límite con técnicas de "*recubrimientos*", con las siguientes características:[3]

- Si una sección particular del programa ya no es necesaria, se carga otra sección desde el almacenamiento secundario ocupando las áreas de memoria liberadas por la sección que ya no se necesita.

- La administración manual por programa del recubrimiento es complicada y dificulta el desarrollo y el mantenimiento.

Protección en los sistemas de un solo usuario

El usuario tiene un completo control sobre la totalidad del almacenamiento principal:

- El almacenamiento se divide en porciones que contienen el S. O., el programa del usuario y una porción sin usar.

[2] Ver Figura 3.2 de la página 69 [7, Deitel].

[3] Ver Figura 3.3 de la página 70 [7, Deitel].

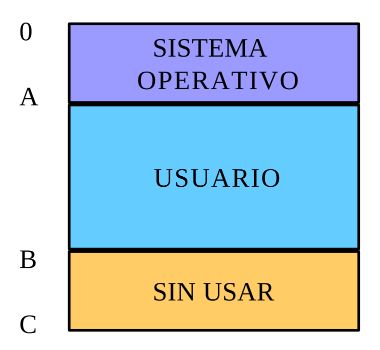


Figura 3.2: Asignación contigua de almacenamiento de un solo usuario.

- El programa del usuario podría destruir áreas del S. O. que podrían:
  - Detener el sistema.
  - Producir salidas erróneas.
- El S. O. debe estar protegido contra el proceso usuario:
  - La protección se instrumenta mediante un "*registro de límites*" incorporado a la cpu:
    - Contiene la dirección de la instrucción más alta utilizada por el S. O.
    - Si se intenta ingresar al S. O. la instrucción es interceptada y el proceso finaliza.

Procesamiento por lotes de flujo único

Los sistemas de un solo usuario se dedican a un trabajo durante más tiempo del que toma su ejecución.

Los trabajos requieren de:

- "*tiempo de instalación*": el necesario para preparar el entorno operativo requerido.
- "*tiempo de descarga*": el necesario para desmontar el entorno operativo que fue requerido.

Durante la instalación y descarga de los trabajos la cpu no está ejecutando dichos trabajos requeridos, por lo cual:

- Automatizar la "*transición de trabajo a trabajo*" reduce la cantidad de tiempo perdido entre trabajos.
- Surgieron los sistemas de "*procesamiento por lotes*".

En el "*procesamiento por lotes de flujo único*" los trabajos se agrupan en "*lotes*" encolándose para su ejecución.

El "*procesador de flujos de trabajos*":

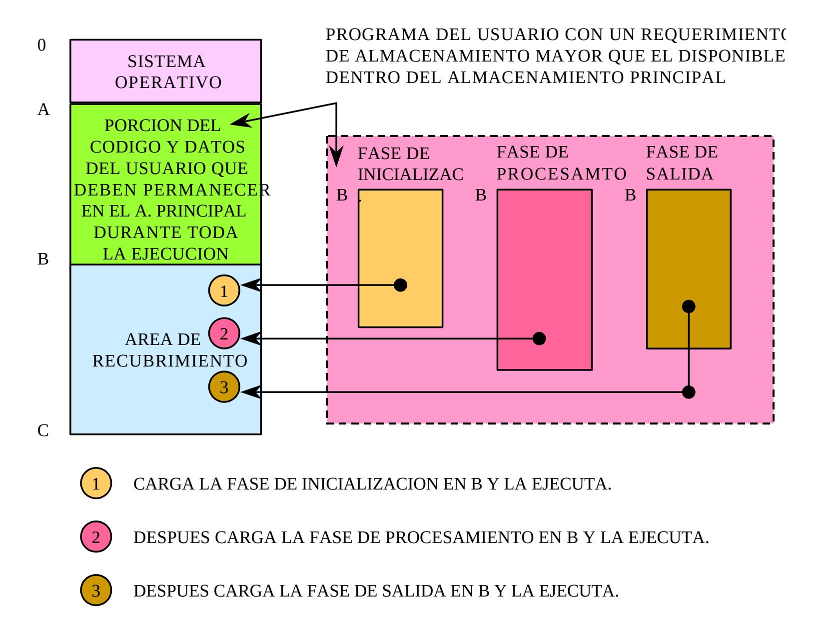


Figura 3.3: Estructura de recubrimiento típica.

- Lee las instrucciones del "*lenguaje de control de trabajos*".
- Facilita la preparación del trabajo siguiente.
- Emite instrucciones al operador del sistema.
- Automatiza funciones anteriormente manuales.
- Cuando finaliza un trabajo efectúa las "*operaciones de mantenimiento*" apropiadas para facilitar la transición del siguiente trabajo.

3.5 Multiprogramación de Partición Fija

Los sistemas de un solo usuario desperdician gran cantidad de recursos computacionales debido a que [7, Deitel]:[4]

- Cuando ocurre una petición de e / s la cpu normalmente no puede continuar el proceso hasta que concluya la operación de e / s requerida.
- Los periféricos de e / s frenan la ejecución de los procesos ya que comparativamente la cpu es varios órdenes de magnitud más rápida que los dispositivos de e / s.

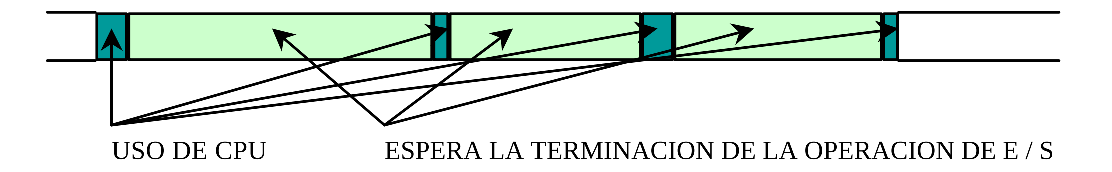
Los sistemas de "*multiprogramación*" permiten que varios procesos usuarios compitan al mismo tiempo por los recursos del sistema:

[4] Ver Figura 3.4 de la página 71 [7, Deitel].

PARA UN USUARIO QUE REALIZA CALCULOS INTENSIVOS:



PARA UN USUARIO QUE REALIZA OPERACIONES REGULARES DE E / S:



NOTA: FRECUENTEMENTE LA LONGITUD DE LAS ESPERAS DE E / S ES MAS GRANDE EN RELACION CON LA LONGITUD DE LOS PERIODOS DE UTILIZACION DE LA CPU DE LO QUE INDICA ESTE DIAGRAMA. SE DEBE A LA RELACION DE VELOCIDADES ENTRE LA CPU Y LOS DISPOSITIVOS DE E / S.

Figura 3.4: Utilización de la cpu en un sistema de un solo usuario.

- Un trabajo en espera de e / s cederá la cpu a otro trabajo que esté listo para efectuar cómputos.
- Existe paralelismo entre el procesamiento y la e / s.
- Se incrementa la utilización de la cpu y la capacidad global de ejecución del sistema.
- Es necesario que varios trabajos residan a la vez en la memoria principal.

3.5.1 Multiprogramación de Partición Fija: Traducción y Carga Absolutas

Las "*particiones*" del almacenamiento principal:

- Son de tamaño fijo.
- Alojan un proceso cada una.
- La cpu se cambia rápidamente entre los procesos creando la **ilusión de simultaneidad**.

Los trabajos se traducían con ensambladores y compiladores absolutos para ser ejecutados solo dentro de una partición específica.[5]

El S. O. resulta de implementación relativamente sencilla pero *no se optimiza la utilización de la memoria.*

[5] Ver Figura 3.5 de la página 72 [7, Deitel].

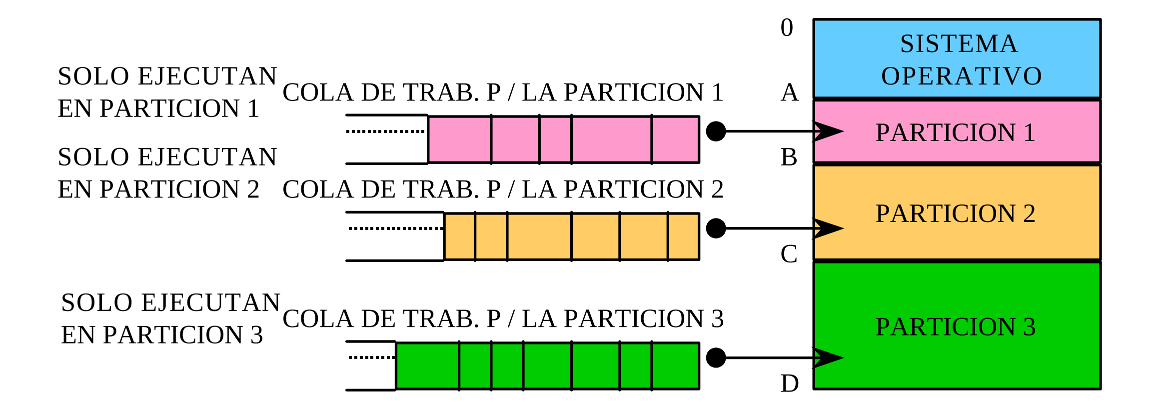


Figura 3.5: Multiprogramación de partición fija con traducción y carga absolutas.

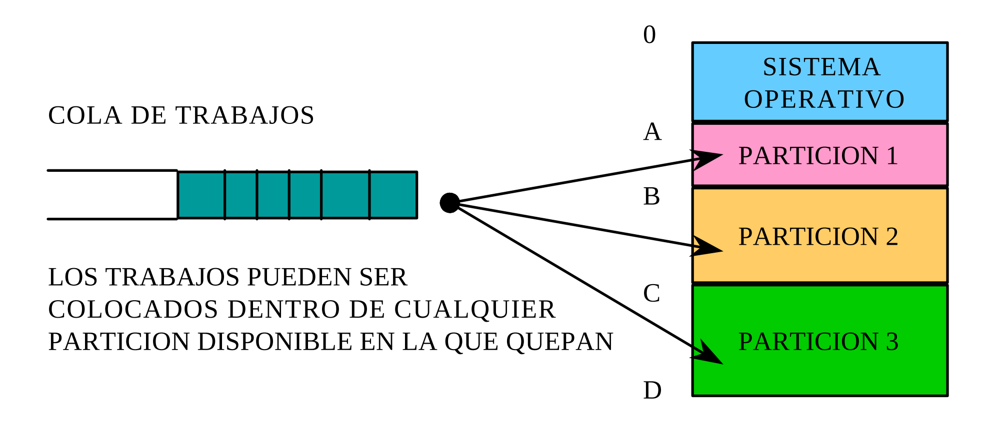


Figura 3.6: Multiprogramación de partición fija con traducción y carga relocalizables.

3.5.2 Multiprogramación de Partición Fija: Traducción y Carga Relocalizables

Los compiladores, ensambladores y cargadores de relocalización:

- Se usan para producir programas relocalizables que puedan ser ejecutados en cualquier partición disponible de tamaño suficiente para aceptarlos.[6]
- Son más complejos que los absolutos.
- Mejoran la utilización del almacenamiento.
- Confieren más flexibilidad en el armado de la carga de procesos.

[6] Ver Figura 3.6 de la página 72 [7, Deitel].

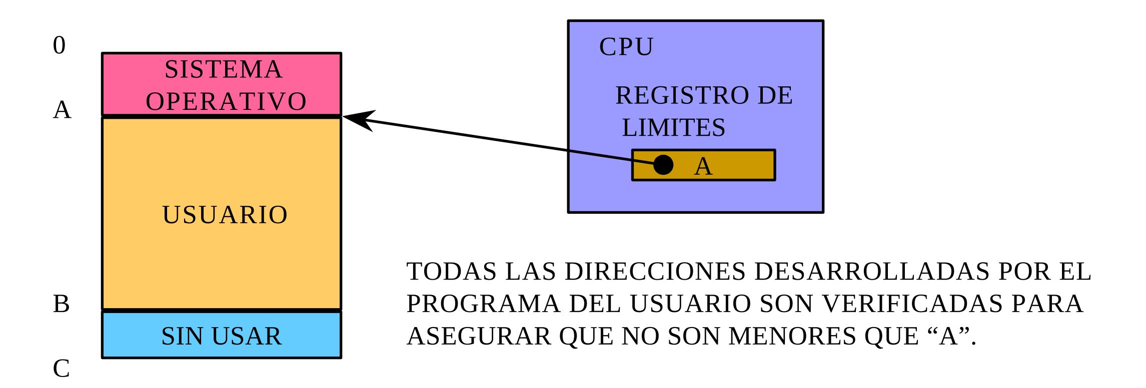


Figura 3.7: Protección del almacenamiento con asignación contigua de un solo proceso de usuario.

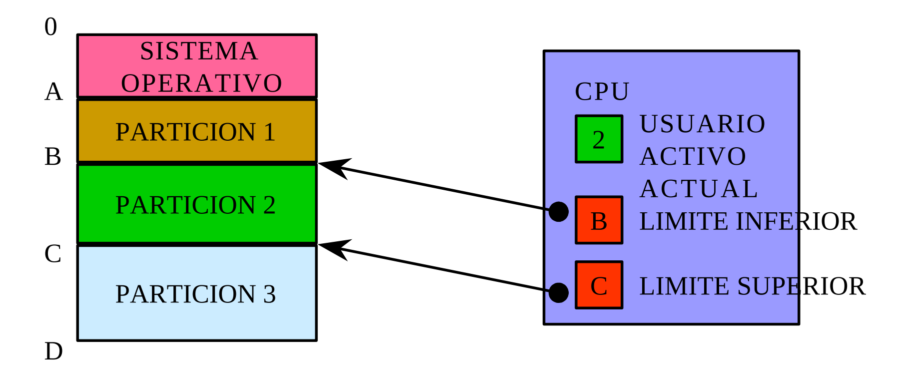


Figura 3.8: Protección del almacenamiento con asignación contigua en sistemas de multiprogramación.

3.5.3 Protección en los Sistemas de Multiprogramación

Si se utiliza asignación contigua de memoria la protección suele implementarse con varios "*registros de límites*".[7]

Los extremos superior e inferior de una partición pueden ser:

- Delineados con dos registros.
- Indicados el límite inferior o superior y el tamaño de la partición o región.

3.5.4 Fragmentación en la Multiprogramación de Partición Fija

La "*fragmentación de almacenamiento*" ocurre en todos los sistemas independientemente de su organización de memoria.

En los S. O. de multiprogramación de partición fija la fragmentación se produce cuando:

[7] Ver Figura 3.7 de la página 73 y Figura 3.8 de la página 73 [7, Deitel].

- Los trabajos del usuario no llenan completamente sus particiones designadas.
- Una partición permanece sin usar porque es demasiado pequeña para alojar un trabajo que está en espera.

3.6 Multiprogramación de Partición Variable

Los procesos ocupan tanto espacio como necesitan, pero obviamente no deben superar el espacio disponible de memoria [7, Deitel].[8]

No hay límites fijos de memoria, es decir que la partición de un trabajo es su propio tamaño.

Se consideran "*esquemas de asignación contigua*", dado que un programa debe ocupar posiciones adyacentes de almacenamiento.

Los procesos que terminan dejan disponibles espacios de memoria principal llamados "*agujeros*":

- Pueden ser usados por otros trabajos que cuando finalizan dejan otros "*agujeros*" menores.
- En sucesivos pasos los "*agujeros*" son cada vez más numerosos pero más pequeños, por lo que se genera un desperdicio de memoria principal.

Combinación de agujeros (áreas libres)

Consiste en *fusionar agujeros adyacentes* para formar uno sencillo más grande.

Se puede hacer cuando un trabajo termina y el almacenamiento que libera tiene límites con otros agujeros.

3.6.1 Compresión o Compactación de Almacenamiento

Puede ocurrir que los agujeros (áreas libres) separados distribuidos por todo el almacenamiento principal constituyan una cantidad importante de memoria:

- Podría ser suficiente (el total global disponible) para alojar a procesos encolados en espera de memoria.
- Podría no ser suficiente ningún área libre individual.[9]

La técnica de *compresión de memoria* implica pasar todas las áreas ocupadas del almacenamiento a uno de los extremos de la memoria principal:

- Deja un solo agujero grande de memoria libre contigua.
- Esta técnica se denomina "*recogida de residuos*".[10]

Principales desventajas de la compresión

Consume recursos del sistema.[11]

[8] Ver Figura 3.9 de la página 75 [7, Deitel].
[9] Ver Figura 3.10 de la página 75 [7, Deitel].
[10] Ver Figura 3.11 de la página 76 [7, Deitel].
[11] Ver Figura 3.12 de la página 76 [7, Deitel].

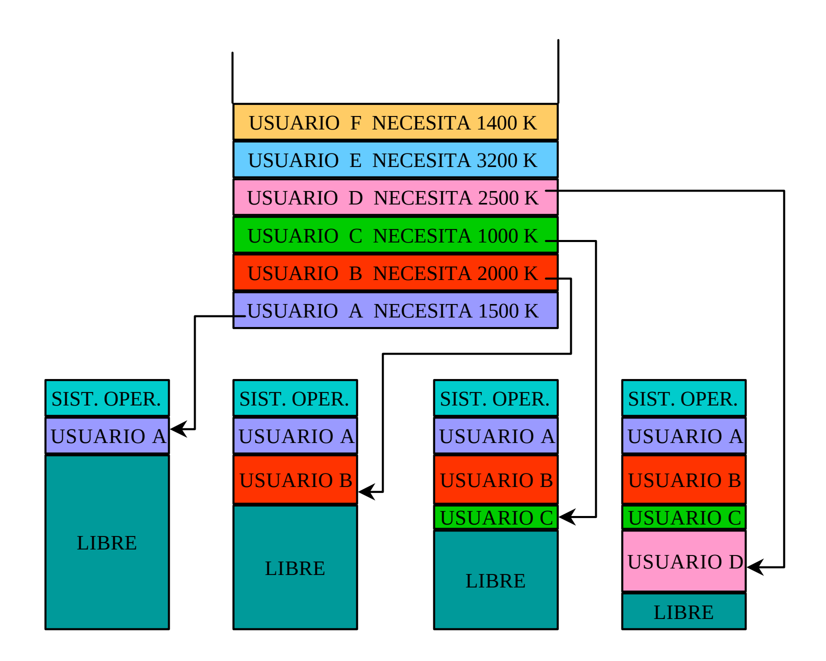


Figura 3.9: Asignación de particiones iniciales en la multiprogramación de partición variable.

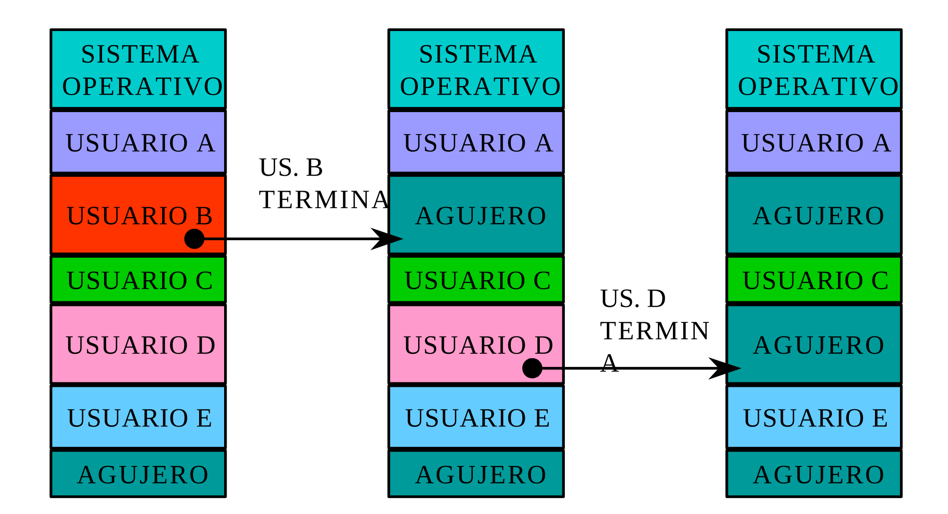


Figura 3.10: Agujeros del almacenamiento en la multiprogramación de partición variable.

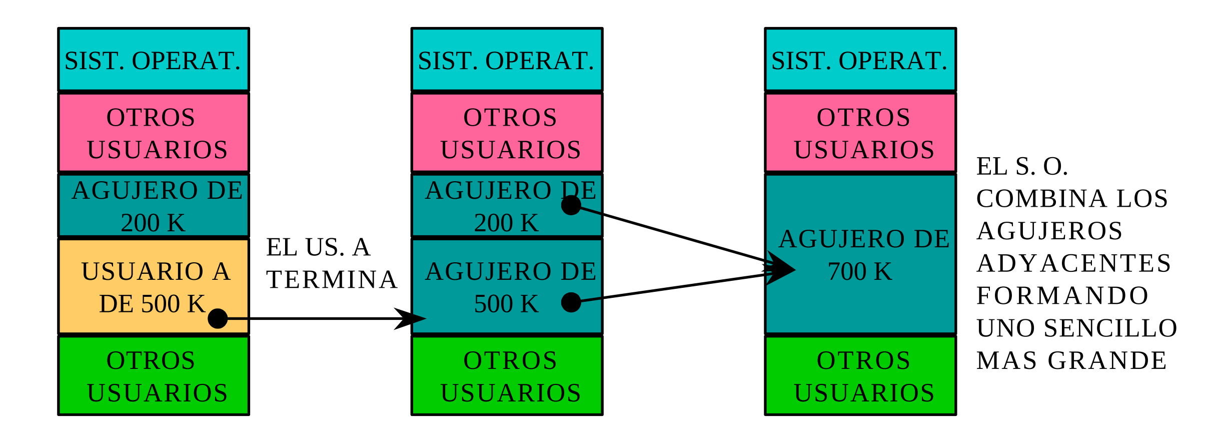
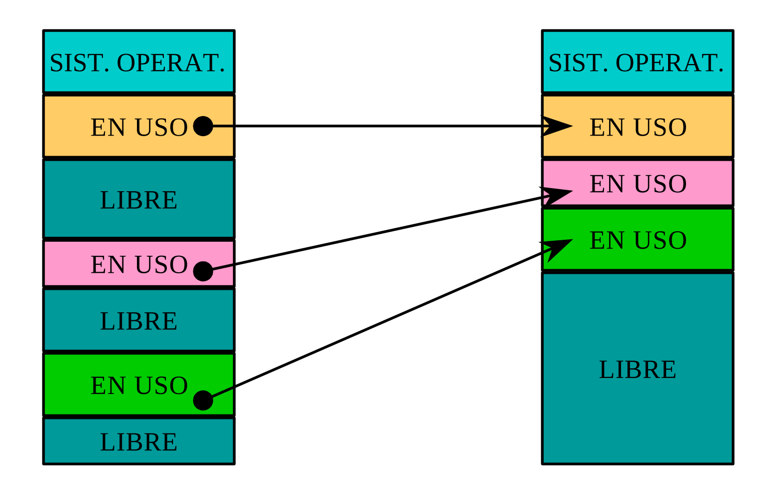


Figura 3.11: Combinación de agujeros adyacentes de almacenamiento en la multiprogramación de partición variable.



EL S. O. COLOCA TODOS LOS BLOQUES “EN USO” JUNTOS, DEJANDO EL ALMACENAMIENTO LIBRE COMO UN UNICO AGUJERO GRANDE.

Figura 3.12: Compresión (compactación) del almacenamiento en la multiprogramación de partición variable.

El sistema debe detener todo mientras efectúa la compresión, lo que puede afectar los tiempos de respuesta.

Implica la relocalización (reubicación) de los procesos que se encuentran en la memoria:

- La información de relocalización debe ser de accesibilidad inmediata.

Una alta carga de trabajo significa mayor frecuencia de compresión que incrementa el uso de recursos.

3.6.2 Estrategias de Colocación del Almacenamiento

Se utilizan para *determinar el lugar de la memoria* donde serán colocados los programas y datos que van llegando y se las clasifica de la siguiente manera:

- "*Estrategia de mejor ajuste*":
  - Un trabajo nuevo es colocado en el agujero en el cual quepa de forma más ajustada:
    - Debe dejarse el menor espacio sin usar.
- "*Estrategia de primer ajuste*":
  - Un trabajo nuevo es colocado en el primer agujero disponible con tamaño suficiente para alojarlo.
- "*Estrategia de peor ajuste*":
  - Consiste en colocar un programa en el agujero en el que quepa de la peor manera, es decir en el más grande posible:
    - El agujero restante es también grande para poder alojar a un nuevo programa relativamente grande.

3.7 Multiprogramación con Intercambio de Almacenamiento

En el esquema de "*intercambio*" *los programas del usuario no requieren permanecer en la memoria principal hasta su terminación* [7, Deitel].

Una variante consiste en que un trabajo se ejecuta hasta que ya no puede continuar:

- Cede el almacenamiento y la cpu al siguiente trabajo.
- La totalidad del almacenamiento se dedica a un trabajo durante un breve período de tiempo.
- Los trabajos son "*intercambiados*", dándose que un trabajo puede ser intercambiado varias veces antes de llegar a su terminación.

Es un esquema razonable y eficiente para un número relativamente reducido de procesos de usuarios.

Los sistemas de intercambio fueron los predecesores de los sistemas de paginación.

El rendimiento de los sistemas de intercambio mejora al reducir el **tiempo de intercambio**:

- Manteniendo al mismo tiempo varias "*imágenes de usuario o imágenes de memoria*" en la memoria principal.
- Retirando una imagen de usuario de la memoria principal solo cuando es necesario su almacenamiento para una nueva imagen.
- Incrementando la cantidad de memoria principal disponible en el sistema.

Las imágenes de usuario (imágenes de memoria) retiradas del almacenamiento principal se graban en el almacenamiento secundario (discos).

3.8 Introducción a la Organización del Almacenamiento Virtual

"*Almacenamiento virtual*" significa la *capacidad de direccionar un espacio de almacenamiento mucho mayor que el disponible en el almacenamiento primario de determinado sistema de computación* [7, Deitel].

Esta tecnología apareció en 1960 en la Universidad de Manchester (Inglaterra), en el sistema "Atlas".

Los **métodos** más comunes de implementación son mediante:

- Técnicas de "*paginación*".
- Técnicas de "*segmentación*".
- Una combinación de ambas técnicas.

Las direcciones generadas por los programas en su ejecución no son, necesariamente, aquellas contenidas en el almacenamiento primario (memoria real), ya que las **direcciones virtuales** suelen seleccionarse dentro de un número mucho mayor de direcciones que las disponibles dentro del almacenamiento primario.

La **evolución en las organizaciones de almacenamiento** puede resumirse como sigue:

- Real:
  - Sistemas dedicados a un solo usuario.
- Real:
  - Sistemas de multiprogramación en memoria real:
    * Multiprogramación en partición fija:

· Absoluta.
· Relocalizable (reubicable).

* Multiprogramación en partición variable.

- Virtual:
  - Multiprogramación en almacenamiento virtual:
    * Paginación pura.
    * Segmentación pura.
    * Combinación paginación / segmentación.

3.9 Conceptos Básicos de Almacenamiento Virtual

La clave del concepto de memoria (almacenamiento) virtual esta en la disociación:

- De las direcciones a las que hace referencia un programa.
- De las direcciones disponibles en la memoria real (almacenamiento primario).

Los principales conceptos son los siguientes:

- "*Direcciones virtuales*":
  - Son las referidas por un proceso en ejecución.
- "*Direcciones reales*":
  - Son las disponibles dentro del almacenamiento primario.
- "*Espacio de direcciones virtuales (v)*" de un proceso:
  - Es el número de direcciones virtuales a que puede hacer referencia el proceso.
- "*Espacio de direcciones reales (r)*" de un computador:
  - Es el número de direcciones reales disponibles en el ordenador.

Los procesos hacen referencia a direcciones virtuales pero éstas deben ejecutarse en el almacenamiento real:

- Las direcciones virtuales *deben ser transformadas* dentro de las direcciones reales, mientras el proceso está en ejecución.
- La *traducción de direcciones* deberá hacerse rápidamente para no degradar al sistema.

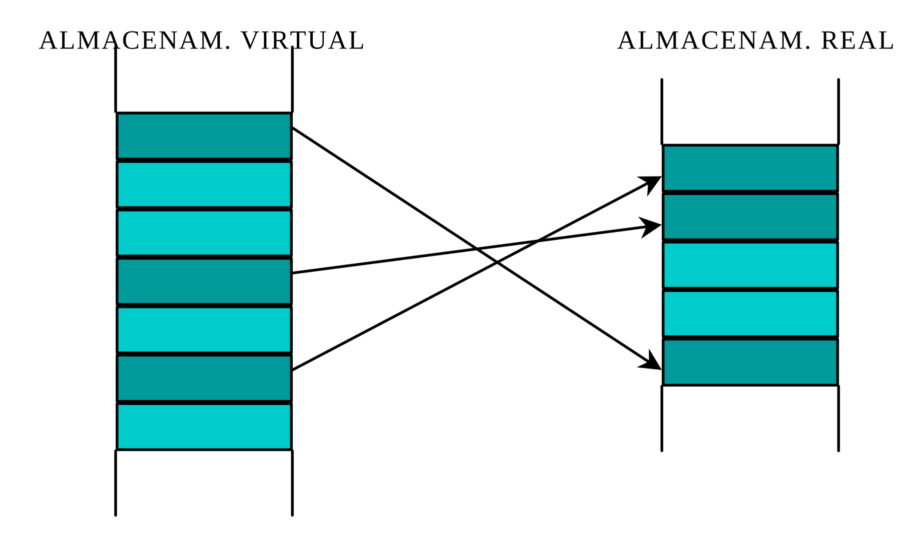


Figura 3.13: Transformación de ítems del espacio de direcciones virtuales al espacio de direcciones reales.

Existen varios medios para asociar las direcciones virtuales con las reales.[12]

Los mecanismos de "*traducción dinámica de direcciones*" (dat) convierten las direcciones virtuales en reales al ejecutarse el proceso.

Las direcciones contiguas dentro del espacio de direcciones virtuales de un proceso no tienen por qué ser contiguas dentro del almacenamiento real, a esto se denomina "*contigüidad artificial*".[13]

3.10 Organización del Almacenamiento de Niveles Múltiples

Se deben proporcionar los medios para retener programas y datos en un gran almacenamiento auxiliar para:

- Permitir que el *espacio de direcciones virtuales* de un usuario sea mayor que el espacio de direcciones reales.
- Soportar *multiprogramación de forma efectiva* en un sistema con muchos usuarios que compartan el almacenamiento real.

Se utiliza un esquema de almacenamiento de dos niveles:[14]

- Primer nivel: "*almacenamiento real*":
  - En él se ejecutan los procesos y en él deben estar los datos para que un proceso pueda referirse a ellos.

[12] Ver Figura 3.13 de la página 80 [7, Deitel].
[13] Ver Figura 3.14 de la página 81 [7, Deitel].
[14] Ver Figura 3.15 de la página 82 [7, Deitel].

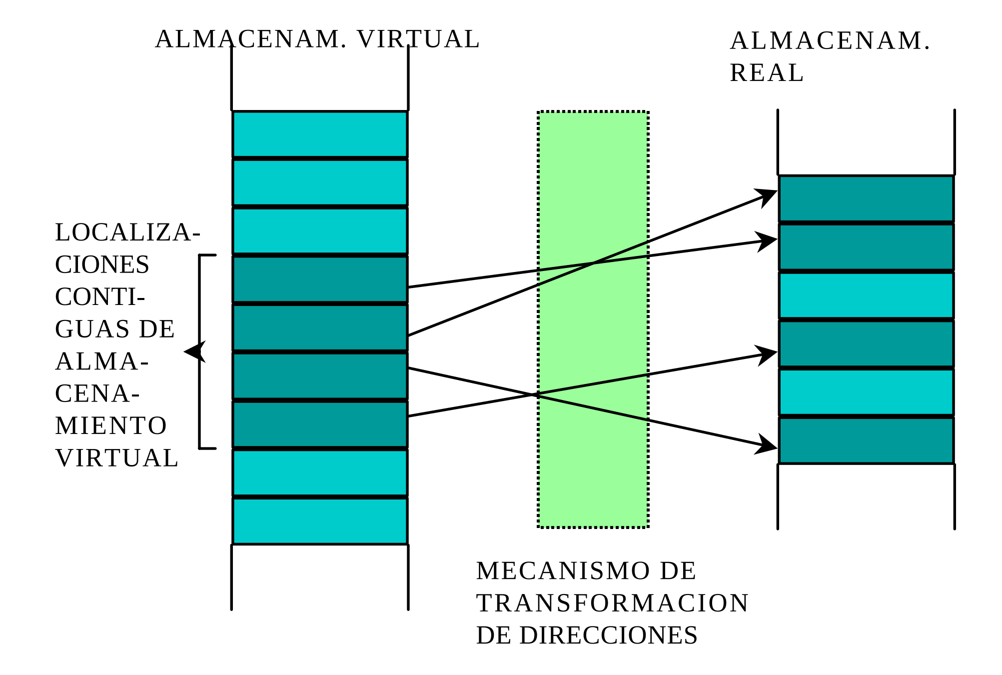


Figura 3.14: Contigüidad artificial.

- Segundo nivel: "*almacenamiento auxiliar, secundario o adicional*":
  - Generalmente consta de discos de gran capacidad que pueden mantener los programas y datos que no caben al mismo tiempo en el más limitado almacenamiento real.

Cuando se va a ejecutar un proceso su código y datos se pasan al almacenamiento principal.

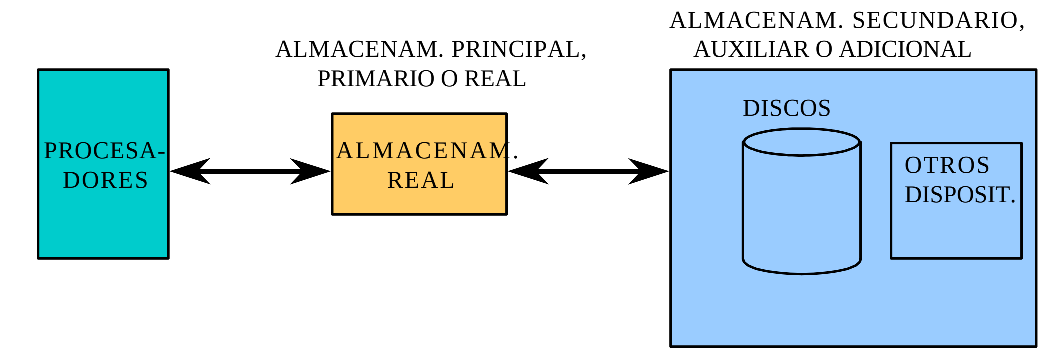
El almacenamiento real es **compartido** por varios procesos:

- Cada proceso puede tener un espacio de direcciones virtuales mucho mayor que el almacenamiento real.
- Solo se mantiene al mismo tiempo una pequeña parte de los programas y datos de cada proceso en el almacenamiento real.

3.11 Transformación de Bloques

Los mecanismos de traducción dinámica de direcciones deben mantener "*mapas*" que ilustren qué direcciones del almacenamiento virtual se encuentran en el almacenamiento real y dónde se encuentran [7, Deitel].

La información se agrupa en "*bloques*":



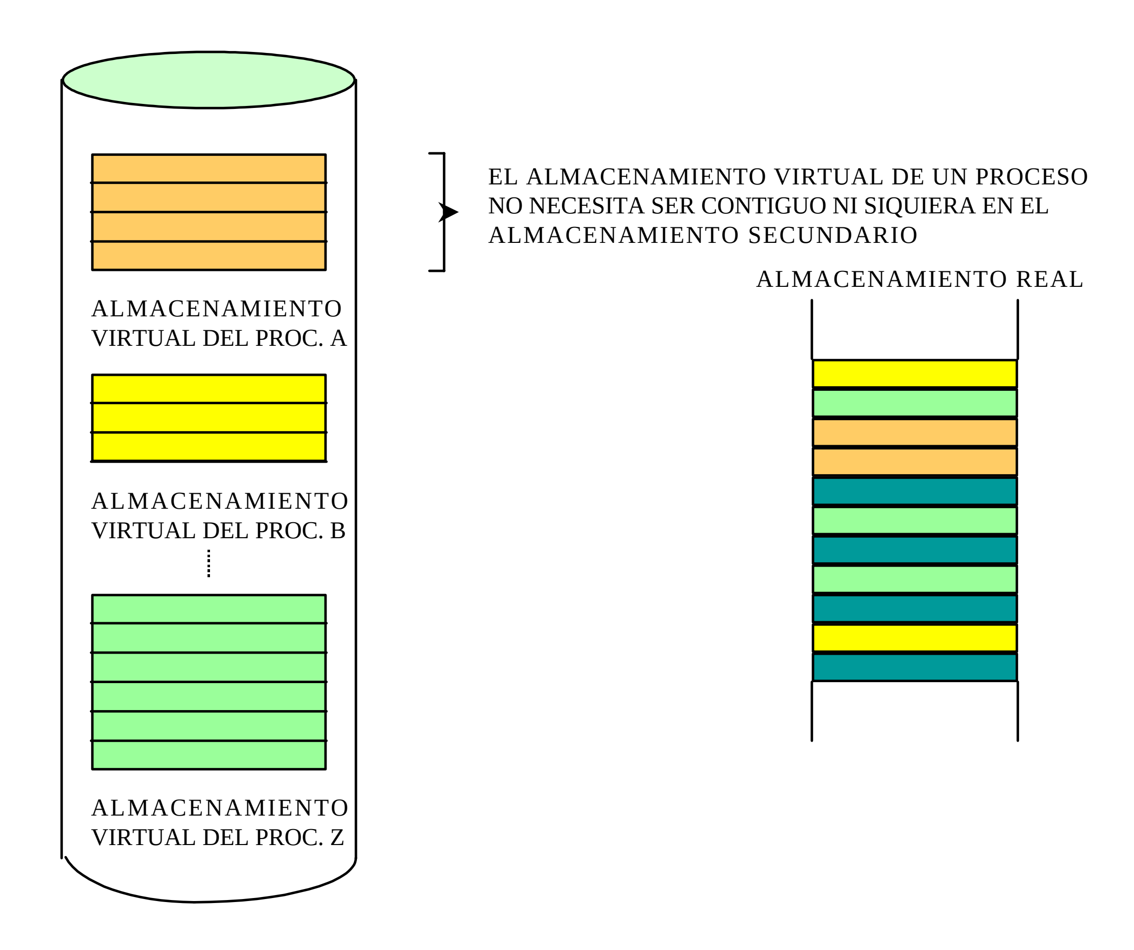


Figura 3.15: Almacenamiento de dos niveles.

- El sistema está informado del lugar del almacenamiento real donde han sido colocados los bloques de almacenamiento virtual.
- Cuanto mayor sea el bloque menor será la fracción del almacenamiento real que debe dedicarse a contener la información del mapa.
- Con bloques grandes:
  - Se reduce la sobrecarga de almacenamiento del mecanismo de transformación.
  - Se incrementa el tiempo de transferencia entre los almacenamientos secundario y primario.
  - Consumen más almacenamiento real pudiendo limitar el número de procesos que pueden compartirlo.
- Los bloques pueden ser de *tamaño*:
  - *Igual*: se denominan "*páginas*" y la organización de almacenamiento virtual asociada se denomina "*paginación*".
  - *Diferente*: se denominan "*segmentos*" y la organización de almacenamiento virtual asociada se denomina "*segmentación*".
- Se pueden combinar ambas técnicas: segmentos de tamaño variable compuestos de páginas de tamaño fijo.

Las direcciones son "*bidimensionales*", es decir que una dirección virtual "*v*" se indica por un par ordenado "*(b,d)*", donde:

- "b": número del bloque donde reside.
- "d": desplazamiento a partir del inicio del bloque.

La traducción de una dirección virtual "v = (b,d)" a la dirección real "r" considera lo siguiente:[15]

- Cada proceso tiene su "*tabla de mapa de bloques*" mantenida por el sistema en el almacenamiento real.
- Un registro especial del procesador llamado "*registro origen de la tabla de bloques*" se carga con la dirección real "a" de la "tabla de mapa de bloques":
  - Contiene una entrada para cada bloque del proceso.
  - Las entradas se mantienen en orden secuencial para el bloque 0, bloque 1, etc.
  - Se añade el bloque número "b" a la dirección base "a" de la "tabla de bloques" para formar la dirección real de la entrada de la "tabla de mapa de bloques" para el bloque "b":
    - Contiene la dirección real "b ' " para el bloque "b".

[15] Ver Figura 3.16 de la página 84 y Figura 3.17 de la página 85 [7, Deitel].



Figura 3.16: Formato de la dirección virtual dentro de un sistema de transformación de bloques.

∗ El desplazamiento "d" se añade a la dirección de inicio del bloque, "b ' ", para formar la "dirección real" deseada: "r = b ' + d".

La transformación de bloques se efectúa en forma dinámica mientras se ejecuta un proceso, por lo cual, si la implementación no es eficiente, su sobrecarga puede causar una degradación del rendimiento que podría eliminar en parte las ventajas de la utilización del almacenamiento virtual.

3.12 Conceptos Básicos de Paginación

Frecuentemente se diferencia entre la "*paginación pura*" y la "*combinación de paginación y segmentación*" [7, Deitel].

Las páginas se transfieren del almacenamiento secundario al primario en bloques llamados "*marcos de páginas*":

- Tienen el mismo tamaño que las páginas.
- Comienzan en direcciones del almacenamiento real que son *múltiplos enteros* del tamaño fijo de la página.
- Podrá colocarse una nueva página dentro de cualquier "*marco de página*" o "*celda de página*" disponible.

La "*traducción dinámica de direcciones*" incluye:

- Un proceso en ejecución hace referencia a una dirección virtual "v = (p,d)".[16]
- Un mecanismo de transformación de páginas busca la página "p" en la "*tabla de páginas*" y determina si la página "p" se encuentra en el marco de página "p ' ".
- La dirección de almacenamiento real se forma por la concatenación de "p ' " y "d".

La tabla de "*mapa de páginas*" debe indicar si se encuentra o no en el almacenamiento primario la página referenciada:

- En caso afirmativo dónde está en la memoria real.
- En caso negativo dónde puede estar en el almacenamiento secundario.

[16]Ver Figura 3.18 de la página 86 [7, Deitel].

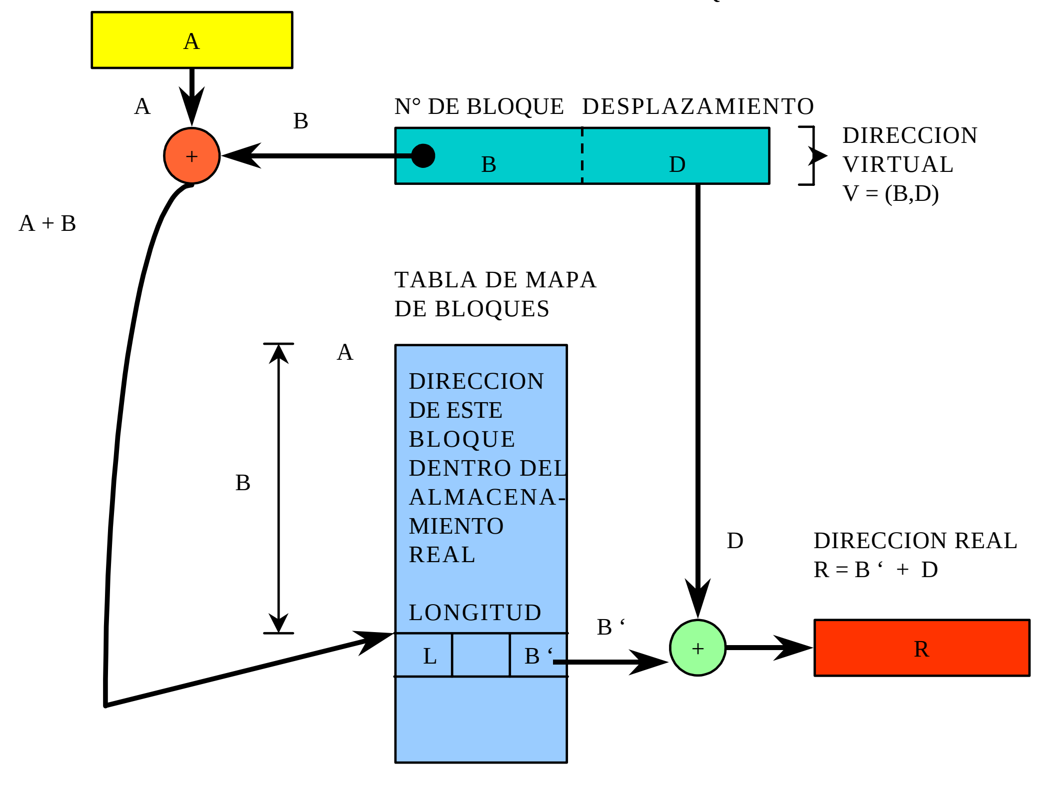


Figura 3.17: Traducción de direcciones virtuales con transformación de bloques.



Figura 3.18: Formato de la dirección virtual en un sistema de paginación pura.

0	MARCO DE PAG. 0
P	MARCO DE PAG. 1
2P	MARCO DE PAG. 2
3P	MARCO DE PAG. 3
4P	MARCO DE PAG. 4
5P	MARCO DE PAG. 5
6P	MARCO DE PAG. 6
7P	MARCO DE PAG. 7

N° DE PAGINA	TAMAÑO DE PAGINA	N° DE DIRECCIONES DE ALMACENAMIENTO REAI
0	P	0 --> P - 1
1	P	P --> 2P - 1
2	P	2P --> 3P - 1
3	P	3P --> 4P - 1
4	P	4P --> 5P - 1
5	P	5P --> 6P - 1
6	P	6P --> 7P - 1
7	P	7P --> 8P - 1
.		
.		
.		
.		

Figura 3.19: Almacenamiento real dividido en marcos de páginas.

La dirección de almacenamiento primario “a”, donde comienza el marco de pagina “p ’ ” (suponiendo un tamaño de página “p”), está dada por: “a = (p) (p ’)”; se supone marcos de página numerados 0, 1, 2, etc.[17]

3.12.1 Traducción de Direcciones de Paginación por Transformación Directa

Un proceso en ejecución hace referencia a la dirección virtual $v = (p,d)$.

Antes que un proceso comience su ejecución, el S. O. carga la dirección de almacenamiento primario de la “*tabla de mapa de páginas*” en el “*registro origen de la tabla de mapa de páginas*”.[18]

La dirección base de la tabla de mapa de páginas es “b”.

El número de página es “p”.

La dirección en el almacenamiento primario de la entrada en la tabla de mapa de páginas para la página “p” es “$b + p$”:

[17] Ver Figura 3.19 de la página 86, Figura 3.20 de la página 87 y Figura 3.21 de la página 88 [7, Deitel].

[18] Ver Figura 3.22 de la página 89 [7, Deitel].

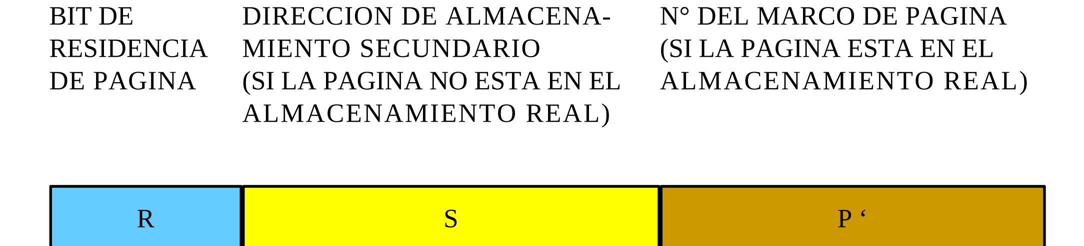


Figura 3.20: Una entrada en la tabla de mapa de páginas.

- Indica que el marco de página "*p '* " corresponde a la página virtual.
- "*p '* " se concatena con el desplazamiento "*d*" par formar la dirección real "*r*".

"*Esto es un ejemplo de transformación directa debido a que la tabla de mapa de páginas contiene una entrada por cada una de las páginas del almacenamiento virtual de este proceso*".

La dirección virtual que se está traduciendo y la dirección base de la tabla de mapa de páginas son mantenidas en un registro de alta velocidad del control del procesador.

La tabla de mapa de páginas transformada directamente suele mantenerse en el almacenamiento primario:

- Las referencias a esta tabla requieren un ciclo completo de almacenamiento primario, que generalmente es la parte más larga de un ciclo de ejecución de instrucciones.
- Se requiere otro ciclo de ejecución de almacenamiento primario para la transformación de páginas, lo que puede ocasionar degradación equivalente a un 50%, para lo cual una solución sería tener la tabla completa de mapa de páginas de transformación directa en la "*caché*" de muy alta velocidad.

3.12.2 Traducción de Direcciones de Paginación por Transformación Asociativa

Una forma de acelerar la traducción dinámica de páginas consiste en colocar la tabla completa de mapa de páginas en un "*almacenamiento asociativo*" que tenga un tiempo de ciclo mucho más rápido que el almacenamiento primario.

Una variante es la "*transformación asociativa pura*".[19]

Un programa en ejecución hace referencia a la dirección virtual $v = (p,d)$.

Cada entrada en el almacenamiento asociativo se busca de forma simultánea para la página "*p*":

[19] Ver Figura 3.23 de la página 90 [7, Deitel].

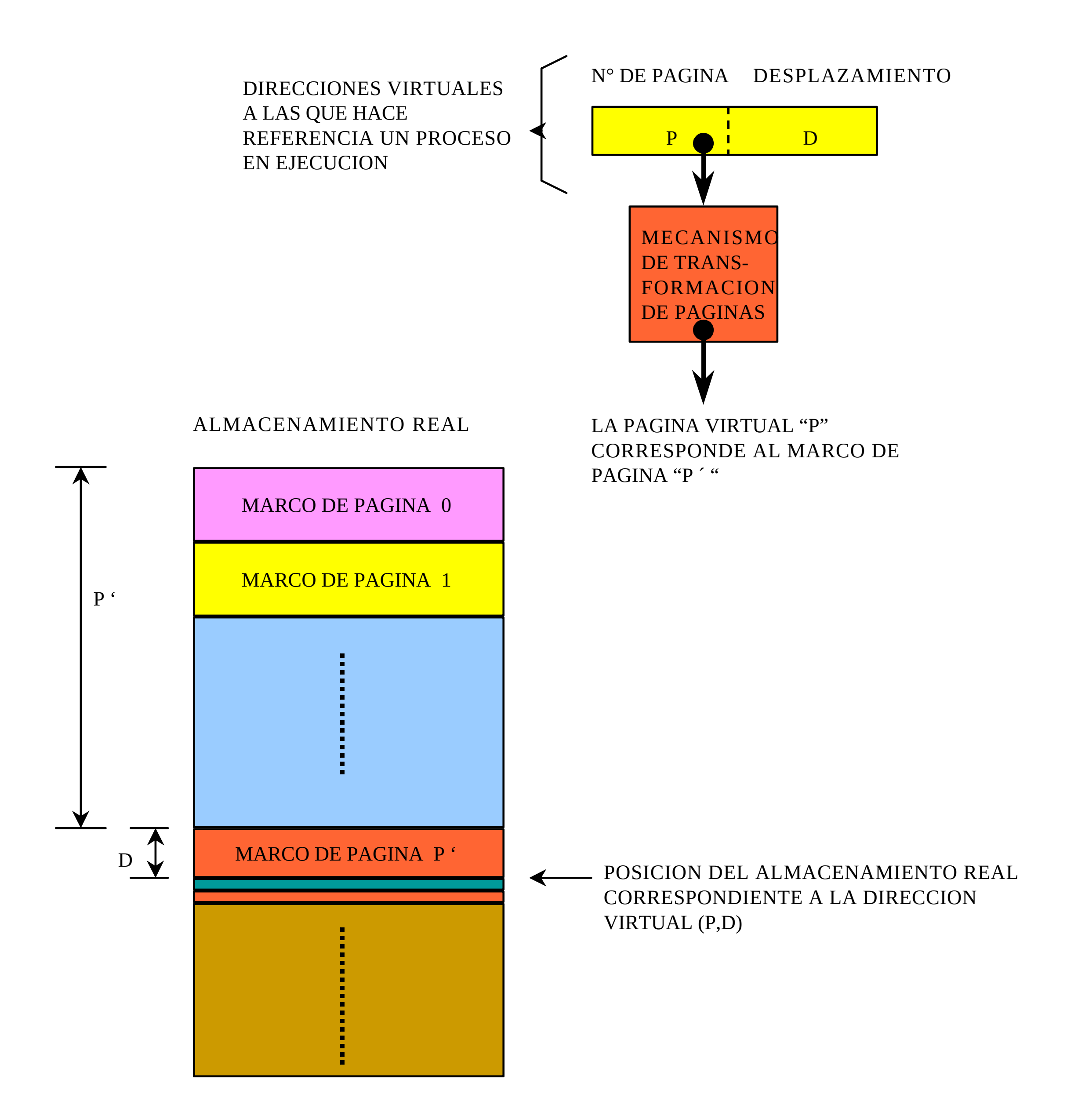


Figura 3.21: Correspondencia entre las direcciones de almacenamiento virtual y las direcciones de almacenamiento real en un sistema de paginación.

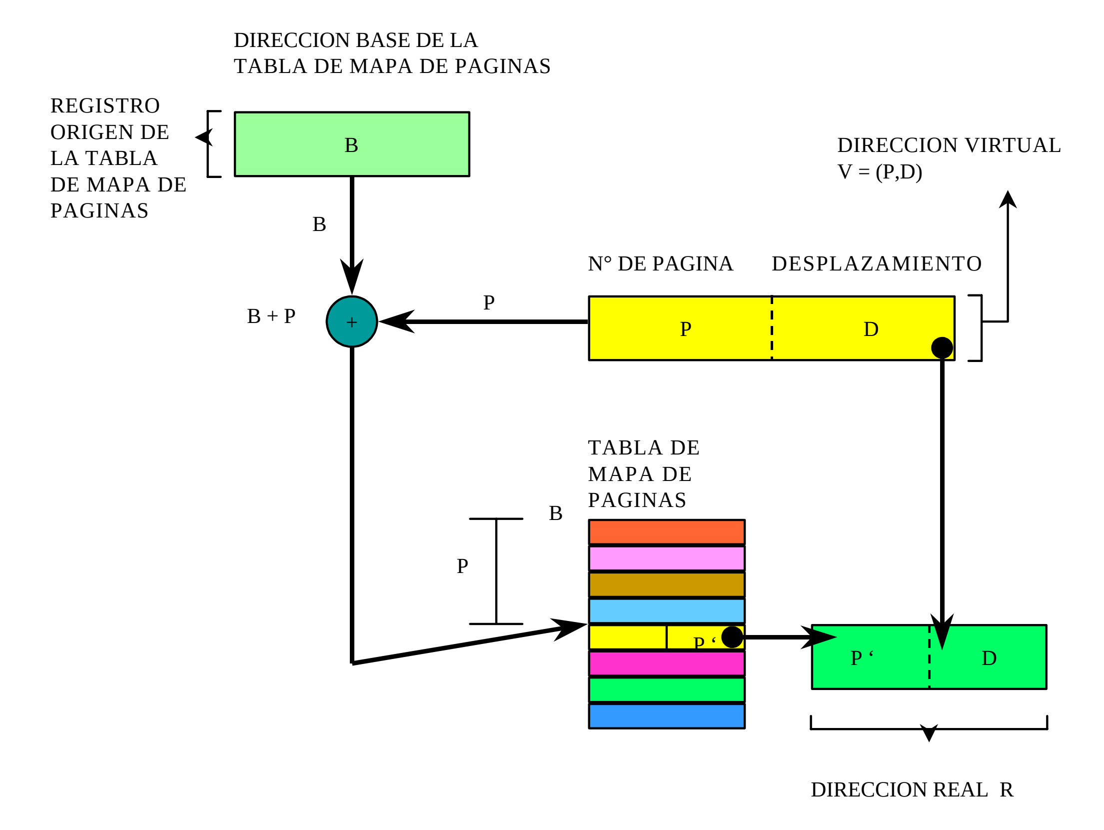


Figura 3.22: Traducción de direcciones de página por transformación directa.

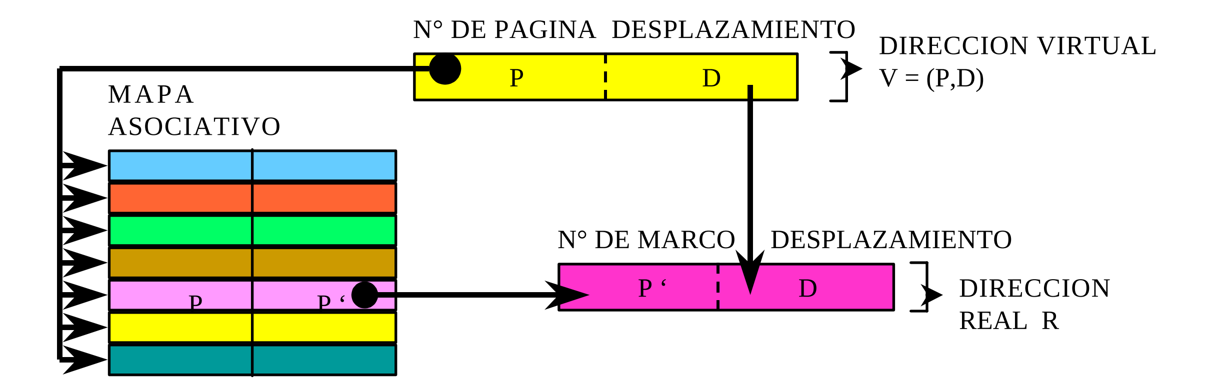


Figura 3.23: Traducción de direcciones de página por transformación asociativa pura.

- Se obtiene "*p ’* " como el marco de página correspondiente a la página "*p*".
- Se concatena "*p ’* " con "*d*" formando la dirección real "*r*".

Cada una de las células del almacenamiento asociativo se registra de manera simultánea:

- Hace costoso el almacenamiento asociativo.
- Implementar la transformación asociativa pura resulta demasiado costoso, tal lo ocurrido con la implementación de la transformación directa pura utilizando "*caché*".

3.12.3 Traducción de Direcciones de Paginación por Combinación de Transformación Asociativa / Directa

Se utiliza un almacenamiento asociativo capaz de mantener solo un pequeño porcentaje del mapa completo de páginas para un proceso.[20]

Las entradas de página contenidas en este *mapa reducido* corresponden solo a las *páginas referenciadas recientemente*:

- Se presupone que una página recientemente referenciada tendrá posibilidades de serlo de nuevo próximamente.
- Los rendimientos obtenidos con este esquema de *mapa asociativo parcial* superan aproximadamente en un 100 % a los rendimientos obtenidos con esquemas de *mapa asociativo de página completo.*

Un programa hace referencia a la dirección virtual $v = (p,d)$.

El mecanismo de traducción de direcciones intenta encontrar la página "*p*" en el mapa de página asociativo parcial:

[20] Ver Figura 3.24 de la página 92 [7, Deitel].

- Si "*p*" se encuentra allí:
  - El mapa asociativo devuelve "*p '* " como el número de marco de página correspondiente a la página virtual "*p*".
  - "*p '* " se concatena con el desplazamiento "*d*" para formar la dirección real "*r*" que corresponde a la dirección virtual $v = (p,d)$.
- Si "*p*" no se encuentra en el mapa de pagina parcial:
  - Se utiliza un mapa directo convencional.
  - La dirección "*b*" del registro de origen de la tabla de páginas se añade a "*p*" para localizar la entrada apropiada a la página "*p*" en la tabla de mapa de páginas de transformación directa del almacenamiento primario.
  - La tabla indica que "*p '* " es el marco de página correspondiente a la página virtual "*p*".
  - "*p '* " se concatena con el desplazamiento "*d*" para formar la dirección real "*r*" correspondiente a la dirección virtual $v = (p,d)$.

3.12.4 Compartimiento de Recursos en un Sistema de Paginación

En sistemas multiprogramados, especialmente en los de tiempo compartido, es común que más de un usuario estén ejecutando los mismos programas:

- Para optimizar el uso de la memoria real se comparten las páginas que pueden ser compartidas:
  - El compartimiento debe ser cuidadosamente controlado para evitar que un proceso modifique datos que otro proceso esta leyendo.[21]
  - Los programas se encuentran divididos en áreas separadas de "*procedimiento*" y "*datos*".
  - Los procedimientos no modificables se llaman "*procedimientos puros reentrantes*".
  - Los datos y procedimientos modificables no pueden ser compartidos.
  - Los datos no modificables (ej.: tablas fijas) son compartibles.
- Se debe identificar cada página como *compartible o no*.
- Habrá marcos (celdas) de páginas compartidos por varios procesos.

El compartimiento:

- Reduce la cantidad de almacenamiento primario necesario para la ejecución eficaz de un grupo de procesos.
- Puede hacer posible que un sistema determinado mantenga una cantidad mayor de usuarios (procesos).

[21] Ver Figura 3.25 de la página 93 [7, Deitel].

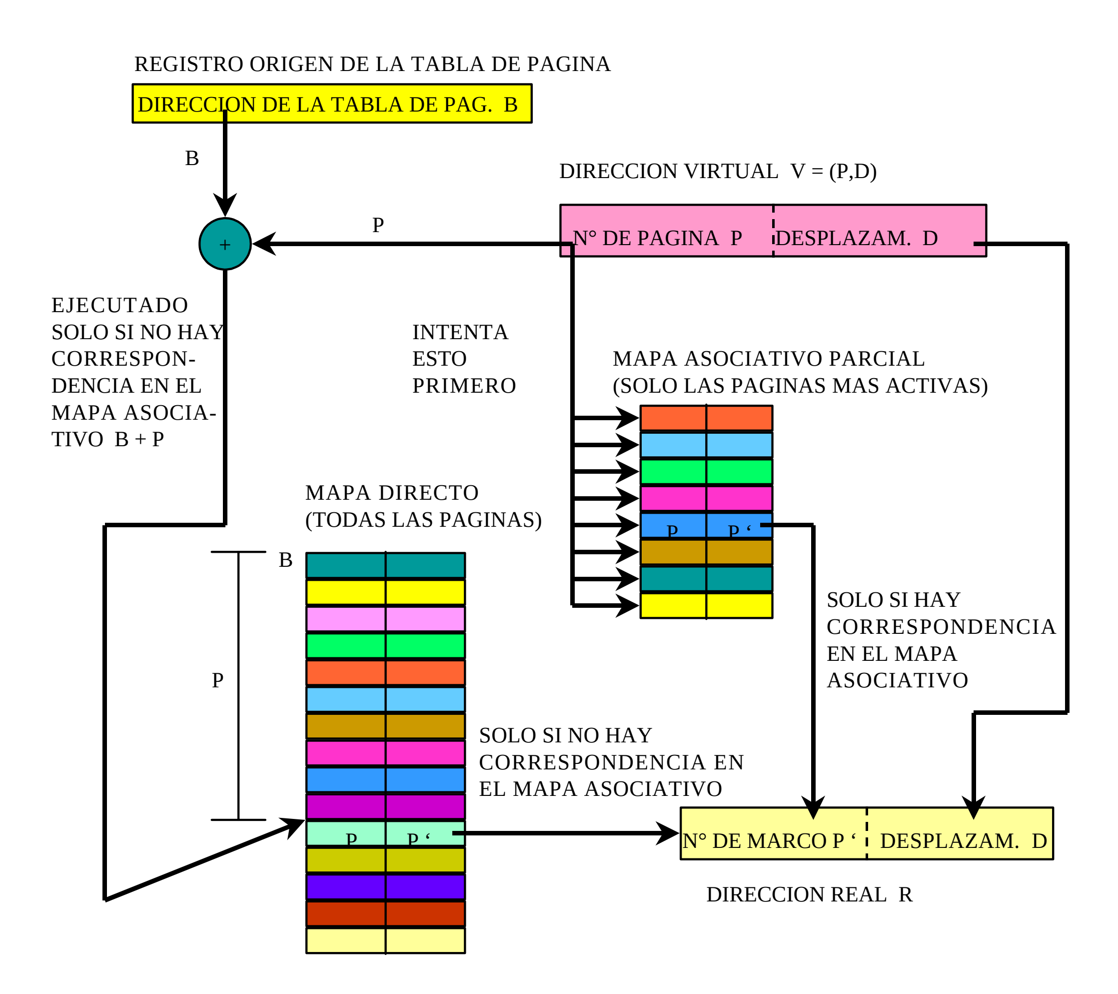


Figura 3.24: Traducción de direcciones de paginación por combinación de transformación asociativa / directa.

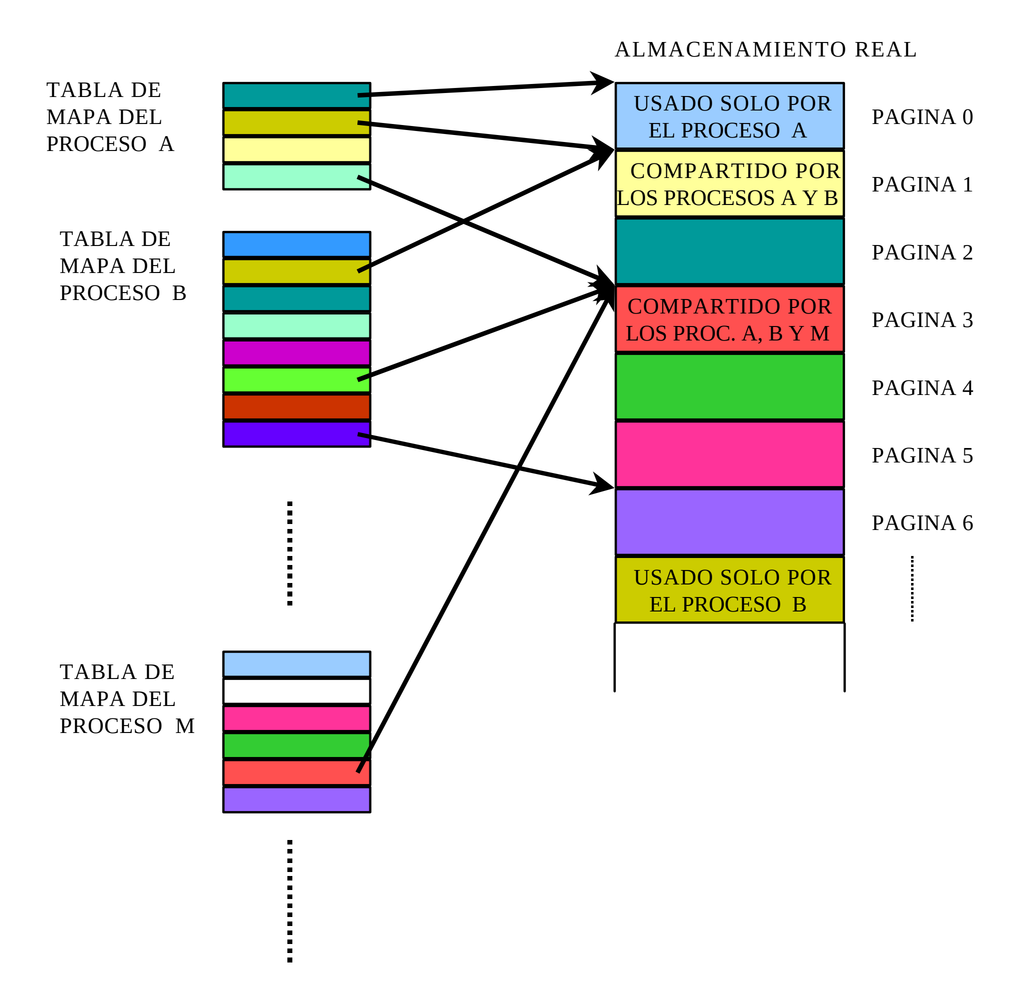


Figura 3.25: Compartimiento en un sistema de paginación pura.

3.13 Segmentación

En los sistemas de "*segmentación*" un programa y sus datos pueden ocupar varios bloques separados de almacenamiento real.[22]

Los *bloques*:

- No necesitan ser de igual tamaño.
- Los bloques separados no necesitan ser adyacentes.
- Deben estar compuestos de posiciones contiguas de almacenamiento.

Se complica la protección de bloques de memoria de un proceso de usuario.

Es más difícil limitar el rango de acceso de cualquier programa [7, Deitel].

Un esquema posible de protección es el uso de *claves de protección del almacenamiento*:[23]

- Las claves están bajo el control estricto del S. O.
- Un programa de usuario, a quien corresponde una cierta clave en la cpu, solo puede hacer referencia a los otros bloques del almacenamiento con **igual** clave de protección.

Una dirección virtual es un par ordenado *v=(s,d)*:[24]

- "*s*" es el número del segmento del almacenamiento virtual en el cual residen los elementos referidos.
- "*d*" es el desplazamiento en el segmento "*s*" en el cual se encuentra el elemento referido.

Un proceso solo puede ejecutarse si su segmento actual (como mínimo) está en el almacenamiento primario.

Los segmentos se transfieren del almacenamiento secundario al primario como *unidades completas.*

Un nuevo segmento puede ser colocado en una serie disponible de posiciones contiguas del almacenamiento primario de tamaño suficiente para alojar al segmento.

La traducción dinámica de direcciones utiliza una "*tabla de mapa de segmentos*".

3.13.1 Control de Acceso en Sistemas de Segmentación

Se le otorga a cada proceso ciertos derechos de acceso a todos los segmentos y se le niega completamente el acceso a muchos otros.

Si un proceso tiene "*acceso de lectura*" a un segmento, puede obtener cualquier elemento de información contenido en ese segmento.

---

[22] Ver Figura 3.26 de la página 95 [7, Deitel].
[23] Ver Figura 3.27 de la página 95 [7, Deitel].
[24] Ver Figura 3.28 de la página 96 [7, Deitel].

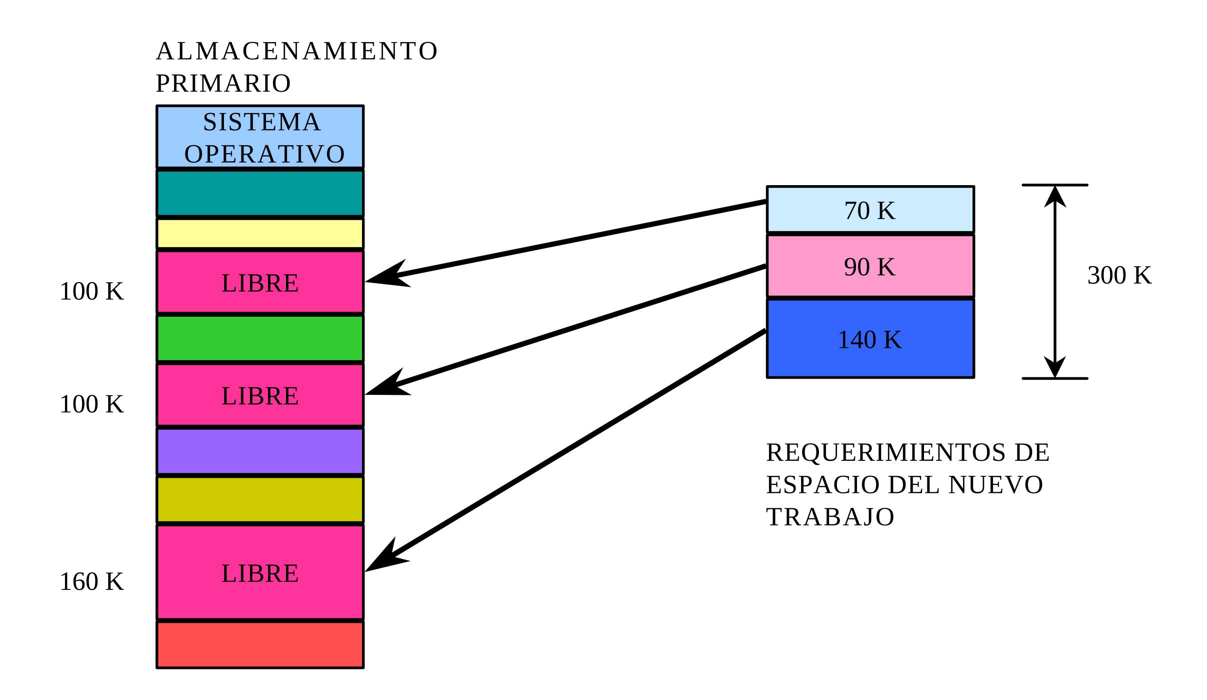


Figura 3.26: Asignación no contigua de almacenamiento.

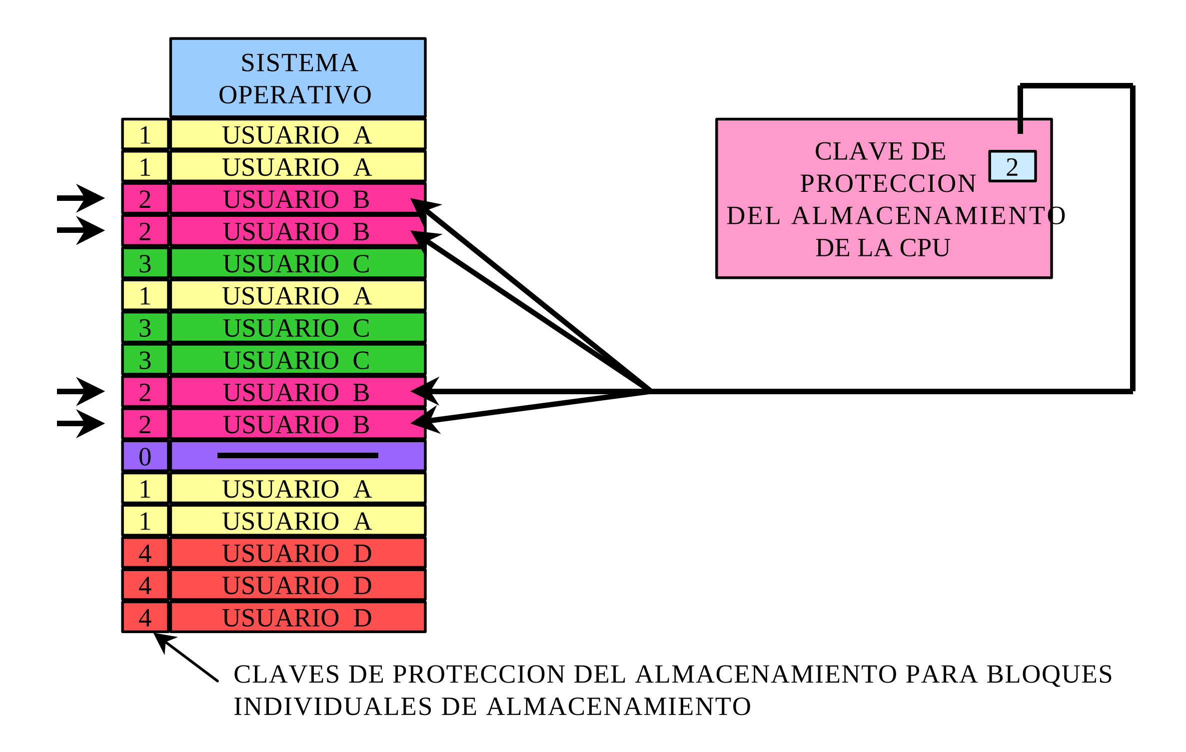


Figura 3.27: Protección del almacenamiento con claves en sistemas de multiprogramación de asignación no contigua de almacenamiento.

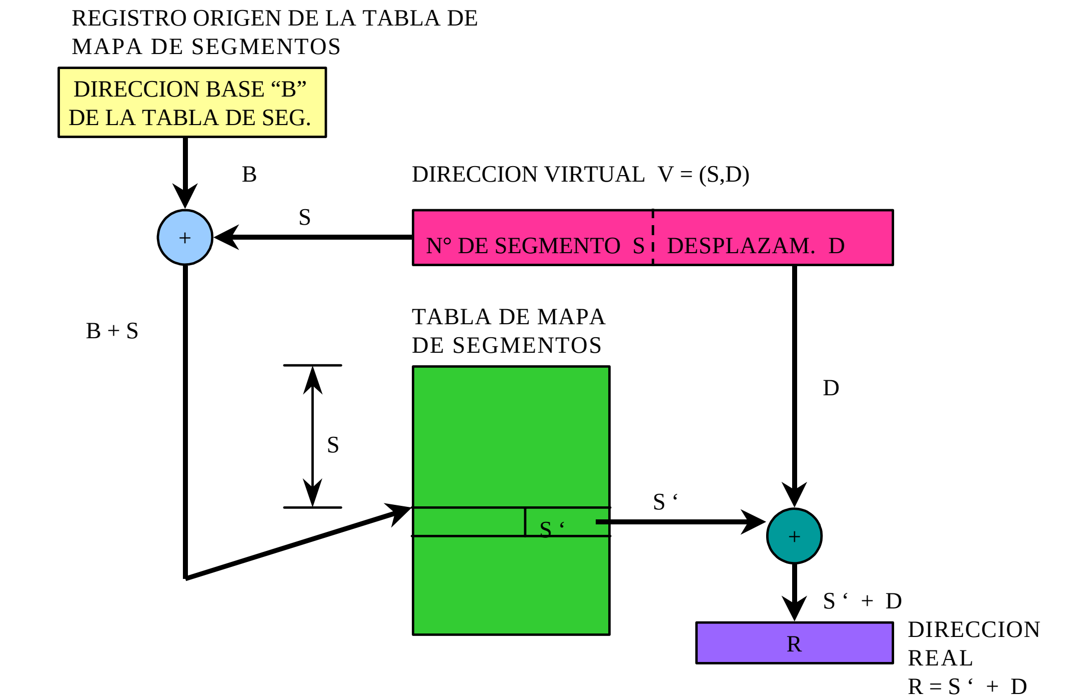


Figura 3.28: Traducción de direcciones virtuales en un sistema de segmentación pura.

Si un proceso tiene "*acceso de escritura*" a un segmento, puede modificar cualquier contenido del segmento y puede introducirle información adicional, incluso destruir toda la información del segmento.

Un proceso con "*acceso de ejecución*" de un segmento puede ejecutarlo como si fuera un programa.

Un proceso con "*acceso de adición*" puede escribir información adicional al final del segmento, pero no puede modificar la información existente.

En base a los "*tipos de control de acceso*" indicados pueden crearse distintos "*modos de control de acceso*".

Ejemplos de combinación de los accesos de lectura, escritura y ejecución para producir modos de protección útiles se dan en la Tabla 3.1 de la página 97 y en la Tabla 3.2 de la página 97 [7, Deitel].

3.13.2 Traducción de Direcciones de Segmentación por Transformación Directa

Existen varias estrategias para la implementación de la traducción de direcciones de segmentación:

- Por transformación directa, asociativa o combinación de asociativa / directa.
- Con caché suficiente para alojar la *tabla completa de mapa de segmentos* o caché parciales que contengan solo las entradas de los *segmentos de referencia más recien-*

Modo	Lectura	Escritura	Ejecución	Explicación
0	N	N	N	No hay permiso de acceso
1	N	N	S	Solo ejecución
2	S	N	N	Solo lectura
3	S	N	S	Lectura / ejecución
4	S	S	N	Lectura / escritura pero no ejecución
5	S	S	S	Acceso no limitado

Tabla 3.1: Ejemplo de combinación de accesos.

Modo	Aplicación
0	Seguridad
1	Un programa disponible a los usuarios, que no pueden copiarlo ni modificarlo, pero sí ejecutarlo
2	Recuperación de información
3	Un programa puede ser copiado o ejecutado, pero no puede ser modificado
4	Protege los datos contra un intento erróneo de ejecutarlos
5	Este acceso se concede a los usuarios de confianza

Tabla 3.2: Ejemplo de aplicaciones de la combinación de accesos.

te.[25]

Se considerará la traducción de direcciones de segmentación con la tabla completa de mapa de segmentos en la caché.

Un proceso en ejecución hace referencia a la dirección virtual $v = (s,d)$:

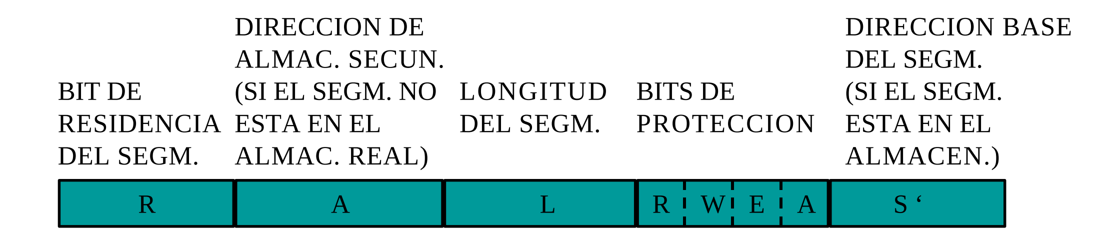
- El segmento número "*s*" se añade a la dirección base "*b*" en el registro origen de la tabla de mapa de segmentos formando la dirección de memoria real "$b + s$", de la entrada para el segmento "*s*" de la *tabla de mapa de segmentos*, que contiene la dirección del almacenamiento primario "*s '* ", donde comienza el segmento.

- El desplazamiento "*d*" se añade a "*s '* " formando la dirección real "$r = d + s'$ ", correspondiente a la dirección virtual "$v = (s,d)$".

Un "*bit de residencia*", "*r*", indica si en la actualidad el segmento se encuentra o no en el almacenamiento primario.

Si el segmento se encuentra en el almacenamiento primario "*s '* " es la dirección en este almacenamiento donde comienza el segmento.

[25] Ver Figura 3.29 de la página 98 [7, Deitel].



R = 0 SI EL SEGMENTO NO ESTA EN EL ALMACENAMIENTO PRIMARIO
R = 1 SI EL SEGMENTO ESTA EN EL ALMACENAMIENTO PRIMARIO

BITS DE PROTECCION: (1-SI, 0-NO)

R - ACCESO DE LECTURA
W - ACCESO DE ESCRITURA
E - ACCESO DE EJECUCION
A - ACCESO DE ADICION

Figura 3.29: Entrada de tabla de mapa de segmentos.

Si el segmento no se encuentra en el almacenamiento primario "*a*" es la dirección en el almacenamiento secundario de donde debe recuperarse antes que el proceso pueda continuar.

Se compara cada referencia a un segmento con los *bits de protección* para determinar si se permite la operación que se está intentando.

Si el segmento buscado no está en el almacenamiento primario se genera un "*fallo de pérdida de segmento*":

- El S. O. obtiene el control y carga el segmento referido desde la dirección "*a*" del almacenamiento secundario.

- Se comprueba si el desplazamiento "*d*" es menor o igual a la longitud del segmento "*l*":

  - Si no es así se genera un "*fallo de desbordamiento de segmento*" y el S. O. obtiene el control y termina la ejecución del proceso.
  - Si el desplazamiento está en el rango del segmento se comprueban los *bits de protección* para asegurarse si se permite la operación que se está intentando:
    - Si es así entonces la dirección base del segmento, "*s ’* ", en el almacenamiento primario se añade al desplazamiento "*d*" formando la dirección de memoria real "$r = s' + d$", que corresponde a la dirección del almacenamiento virtual "$v = (s,d)$".
    - Si la operación intentada no se permite se genera un "*fallo de protección de segmento*" y el S. O. obtiene el control y termina la ejecución del proceso.

3.13.3 Compartimiento en un Sistema de Segmentación

Una de las *ventajas de la segmentación sobre la paginación* es que se trata más de un hecho lógico que físico:

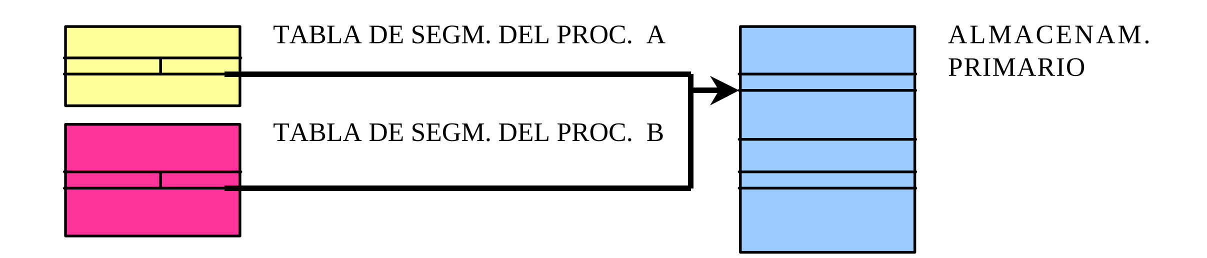


Figura 3.30: Compartimiento en un sistema de segmentación pura.

- En un sistema de segmentación, una vez que un segmento ha sido declarado como compartido, entonces las estructuras que lo integran pueden cambiar de tamaño.

- Lo anterior no cambia el hecho lógico de que residen en un segmento compartido.

Dos procesos pueden compartir un segmento con solo tener entradas en sus tablas generales que apunten al mismo segmento del almacenamiento primario.[26]

3.14 Sistemas de Paginación / Segmentación

Ofrecen las ventajas de las dos técnicas de organización del almacenamiento virtual [7, Deitel].

El tamaño de los segmentos es múltiplo del de las páginas.

No es necesario que todas las páginas de un segmento se encuentren al mismo tiempo en el almacenamiento primario.

Las páginas de almacenamiento virtual, que son contiguas en este almacenamiento, no necesitan ser contiguas en el almacenamiento real.

El *direccionamiento es tridimensional* con una dirección de almacenamiento virtual "*v = (s,p,d)*":

- "*s*" es el número del segmento.

- "*p*" es el número de página.

- "*d*" es el desplazamiento en la página donde se encuentra asignado el elemento deseado.

3.14.1 Traducción Dinámica de Direcciones en Sistemas de Paginación / Segmentación

Se considera la traducción dinámica de direcciones de virtuales a reales en un sistema de paginación / segmentación utilizando la combinación de transformación asociativa / directa.[27]

[26] Ver Figura 3.30 de la página 99 [7, Deitel].
[27] Ver Figura 3.31 de la página 102 [7, Deitel].

El proceso en ejecución hace referencia a la dirección virtual $v = (s,p,d)$.[28]

Las páginas de referencia más reciente tienen entradas en un almacenamiento asociativo.

Se realiza una búsqueda asociativa para intentar localizar *(s,p)* en el almacenamiento asociativo:

- Si se encuentra *(s,p)*, entonces el marco de página "*p '* " en el cual reside dicha página en la memoria real, se concatena al desplazamiento "*d*" para formar la dirección de memoria real "*r*" correspondiente a la dirección virtual $v = (s,p,d)$.

- Si no se encuentra *(s,p)*, entonces:

  - La dirección base "*b*" de la tabla de segmentos se añade al número de segmento "*s*" formando la dirección "$b + s$" de la entrada de la tabla de mapa de segmentos para el segmento "*s*" de la memoria real.
  - La entrada de la tabla de mapa de segmentos indica la dirección base "*s '* " de la tabla de páginas para el segmento "*s*".
  - El número de página "*p*" se añade a "*s '* " formando la dirección "$p + s'$" de la entrada en la tabla de páginas para la página "*p*" del segmento "*s*":
    - Indica que "*p '* " es el número del marco correspondiente a la página virtual "*p*".
    - "*p '* " se concatena con el desplazamiento "*d*" formando la dirección real "*r*" que corresponde a la dirección virtual $v = (s,p,d)$.

Si el segmento "*s*" no se encuentra en el almacenamiento primario se produce un "*fallo de pérdida de segmento*", cuyo caso el S. O. localiza el segmento en el almacenamiento secundario, crea una tabla de páginas para el segmento y carga la página apropiada en el almacenamiento primario, pudiendo producir reemplazos de páginas.

Si el segmento "*s*" está en el almacenamiento primario y si la referencia a la tabla de mapa de páginas indica que la página deseada no se encuentra en el almacenamiento primario, se produce un "*fallo de pérdida de página*", en tal caso el S. O. obtiene el control, localiza la página en el almacenamiento secundario y la carga, pudiendo reemplazar otra página.

Si una dirección de almacenamiento virtual está más allá del final del segmento se genera un "*fallo de desbordamiento de segmento*", el que debe ser atendido por el S. O.

Si los *bits de protección* indican que la operación que se va a ejecutar en la dirección virtual referida no se permite, se genera un "*fallo de protección de segmento*", el que también debe ser atendido por el S. O.

Si se utiliza un mecanismo de transformación directa pura, manteniendo el mapa completo dentro del almacenamiento primario, la referencia promedio de almacenamiento virtual requeriría:

- Un ciclo de almacenamiento para acceder a la *tabla de mapa de segmentos.*

[28] Ver Figura 3.32 de la página 103 [7, Deitel].

- Un segundo ciclo de almacenamiento para hacer referencia a la *tabla de mapa de páginas*.
- Un tercer ciclo de almacenamiento para referenciar al elemento deseado del *almacenamiento real*.

Cada referencia a un elemento comprende tres ciclos de almacenamiento:

- El sistema correría casi a 1 / 3 de su velocidad nominal.
- La traducción de direcciones insumiría 2 / 3 del tiempo.

Con la utilización de registros asociativos (por ej. 16 registros), se logran velocidades de ejecución del 90 % o más de la velocidad total de procesamiento de sus procesadores de control.

La estructura de tablas de procesos, de mapas de segmentos y de mapas de páginas puede consumir un porcentaje importante del almacenamiento primario cuando se ejecutan un gran número de procesos.

La traducción procede mucho más rápido si todas las tablas están en el almacenamiento primario, lo que resta espacio para los procesos.

3.14.2 Compartimiento en un Sistema de Paginación / Segmentación

Se implementa disponiendo entradas en tablas de mapa de segmentos para diferentes procesos que apunten a la misma tabla de *mapa de páginas*.[29]

El compartimiento requiere una administración cuidadosa por parte del S. O., ya sea en sistemas de paginación, segmentación o paginación / segmentación, pues se debe considerar qué sucedería si una nueva página reemplazara a otra página compartida por muchos procesos.

3.15 Estrategias de Administración del Almacenamiento Virtual

Las diferentes **organizaciones de almacenamiento virtual** generalmente implementadas son [7, Deitel]:

- Paginación.
- Segmentación.
- Segmentación y paginación.

Las estrategias para la administración de sistemas de almacenamiento virtual condicionan la conducta de los sistemas de almacenamiento virtual que operan según esas estrategias.

Se consideran las siguientes estrategias:

[29] Ver Figura 3.33 de la página 104 [7, Deitel].

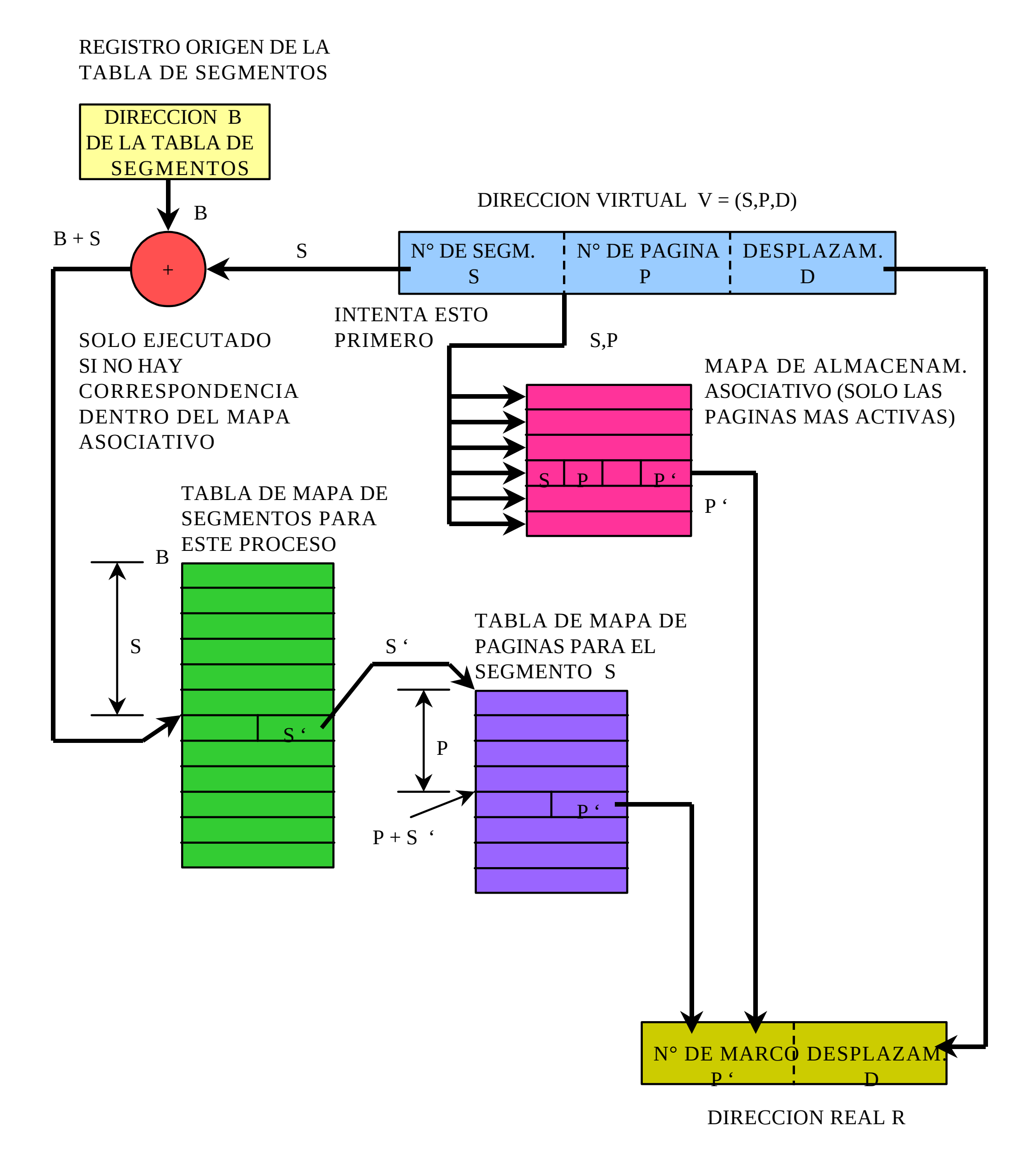


Figura 3.31: Traducción de direcciones virtuales con combinación de transformación asociativa / directa dentro de un sistema de paginación y segmentación.

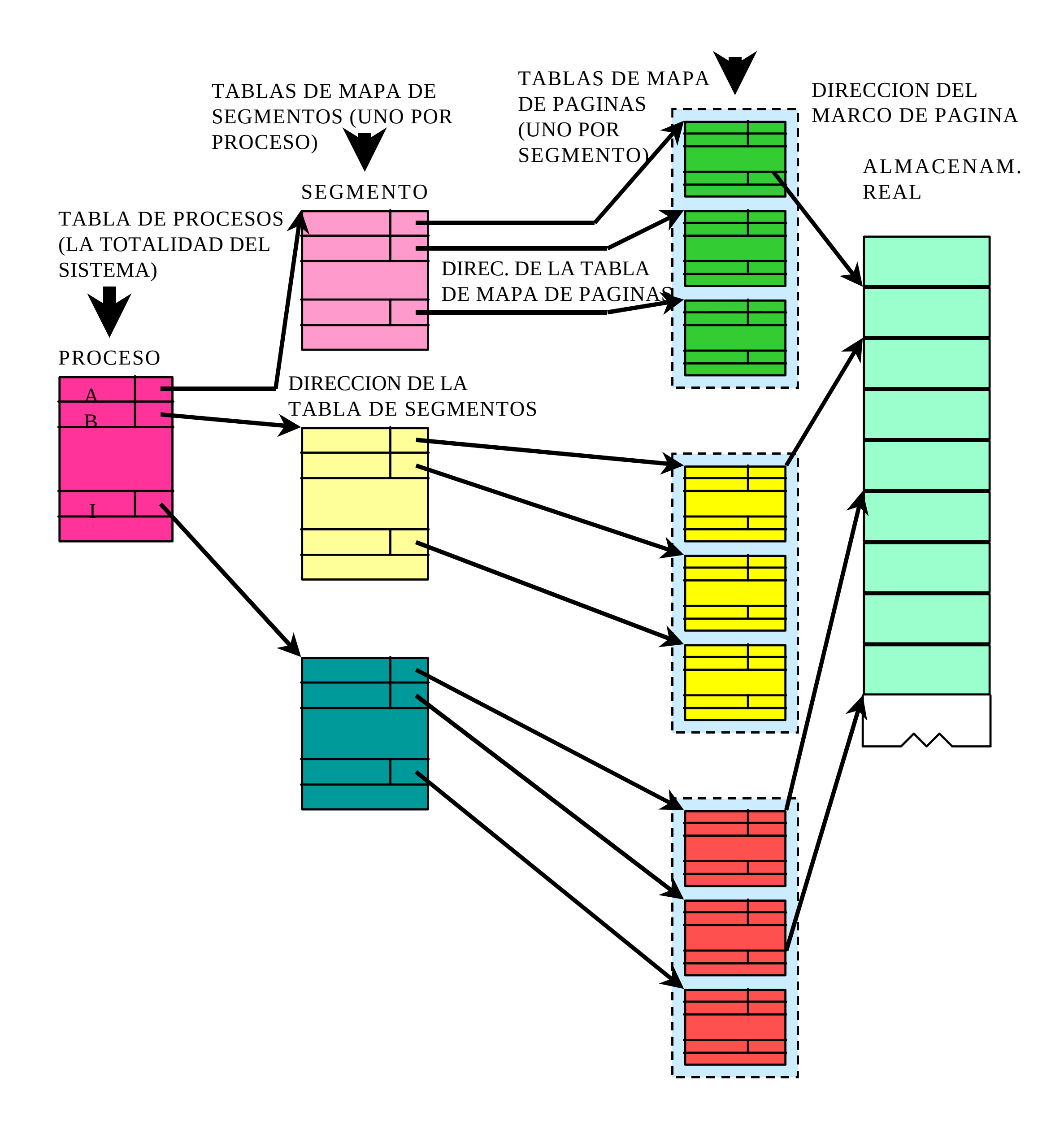


Figura 3.32: Estructura de tablas para un sistema de paginación y segmentación.

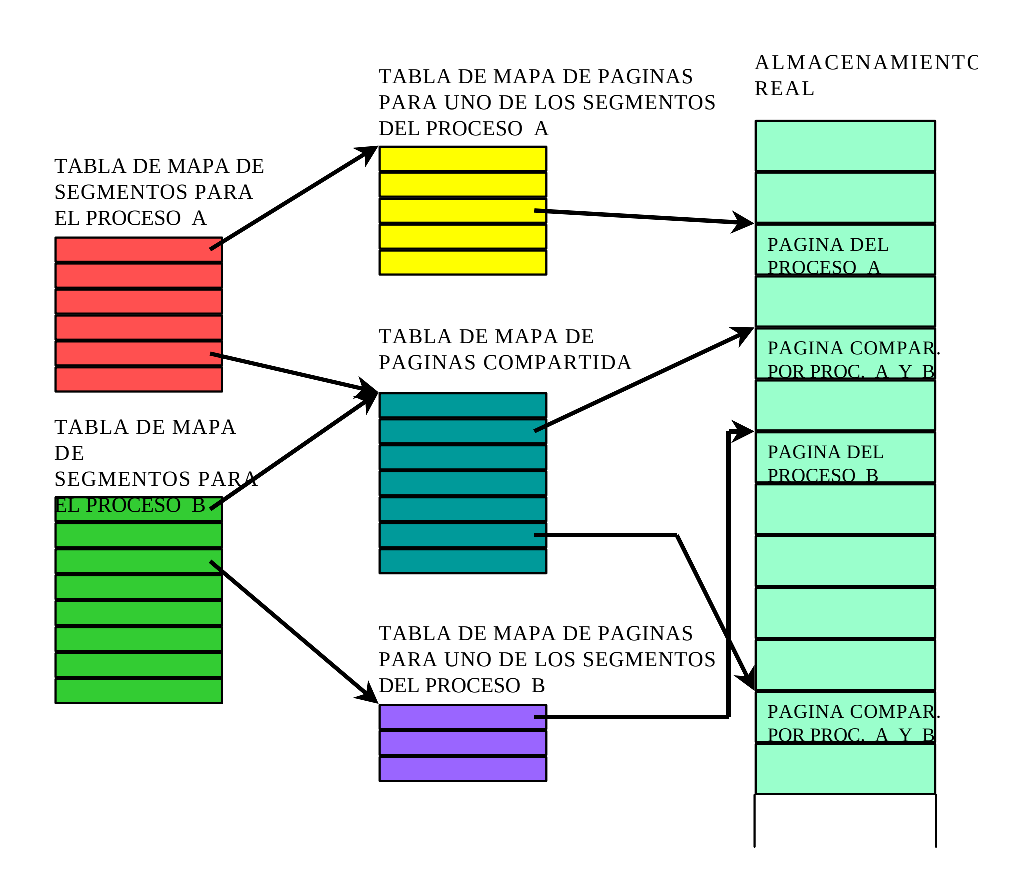


Figura 3.33: Dos procesos compartiendo un sistema de paginación y segmentación.

- "*Estrategias de búsqueda*":
  - Tratan de los casos en que una página o segmento deben ser traídos del almacenamiento secundario al primario.
  - Las estrategias de "*búsqueda por demanda*" esperan a que se haga referencia a una página o segmento por un proceso antes de traerlos al almacenamiento primario.
  - Los esquemas de "*búsqueda anticipada*" intentan determinar por adelantado a qué páginas o segmentos hará referencia un proceso para traerlos al almacenamiento primario antes de ser explícitamente referenciados.
- "*Estrategias de colocación*":
  - Tratan del lugar del almacenamiento primario donde se colocará una nueva página o segmento.
  - Los sistemas toman las decisiones de colocación de una forma trivial ya que una nueva página puede ser colocada dentro de *cualquier* marco de página disponible.
- "*Estrategias de reposición*":
  - Tratan de la decisión de cuál página o segmento desplazar para hacer sitio a una nueva página o segmento cuando el almacenamiento primario está completamente comprometido.

3.15.1 Estrategias de Reposición de Página

Las principales son:

- El principio de optimización.
- Reposición de páginas al azar.
- Primero en entrar - primero en salir.
- Menos recientemente usada.
- Menos frecuentemente usada.
- No usada recientemente.
- Conjuntos de trabajo.

3.15.2 El Principio de Optimización

El "*principio de optimización*" indica que para obtener un rendimiento óptimo, la página que se va a reponer es una que no se va a utilizar en el futuro durante el período de tiempo más largo.

El problema es que no es factible predecir el futuro.

3.15.3 Reposición de Página al Azar

Consiste en escoger al azar la página que va a ser reemplazada.

Todas las páginas del almacenamiento principal deben tener **la misma probabilidad** de ser reemplazadas.

Debe poder seleccionar cualquier página, incluyendo la que va a ser referenciada a continuación (peor selección).

Este esquema es raramente usado.

3.15.4 Reposición de Página por el Sistema de Primero en Entrar - Primero en Salir (FIFO)

Se registra el momento en que cada página ingresa al almacenamiento primario.

Para reemplazar una página, se selecciona aquella que ha estado más tiempo almacenada.

Se presenta el inconveniente de que se pueden reemplazar páginas muy usadas, que serán llamadas de nuevo al almacenamiento primario casi de inmediato.

Se puede presentar la llamada "*anomalía FIFO*":

- Belady, Nelson y Shedler descubrieron que con la reposición FIFO, ciertos patrones de referencias de páginas causan más fallos de páginas cuando se *aumenta* el número de marcos (celdas) de páginas asignados a un proceso: en esto consiste la "*anomalía FIFO*".
- **Esta anomalía contradice a la intuición.**[30]

3.15.5 Reposición de Página Menos - Recientemente - Usada (LRU)

Esta estrategia selecciona para ser reemplazada la página que no ha sido usada durante el mayor período de tiempo.

Se basa en la heurística de que el pasado reciente es un buen indicador del futuro próximo.

Requiere que cada página reciba un "*sello de tiempo*" cada vez que se referencia:

- Puede significar una sobrecarga adicional importante.
- No se implementa frecuentemente.

La página seleccionada para reemplazo podría ser la próxima en ser requerida, por lo que habría que paginarla de nuevo al almacenamiento principal casi de inmediato.

3.15.6 Reposición de Página Menos - Frecuentemente - Usada (LFU)

Acá interesa la **intensidad de uso** que haya tenido cada página.

La página que será reemplazada es aquella que ha sido usada con menos frecuencia o que ha sido referida con menos intensidad.

El *inconveniente* es que se puede seleccionar fácilmente para su reposición la página equivocada:

[30] Ver Figura 3.34 de la página 107 [7, Deitel].

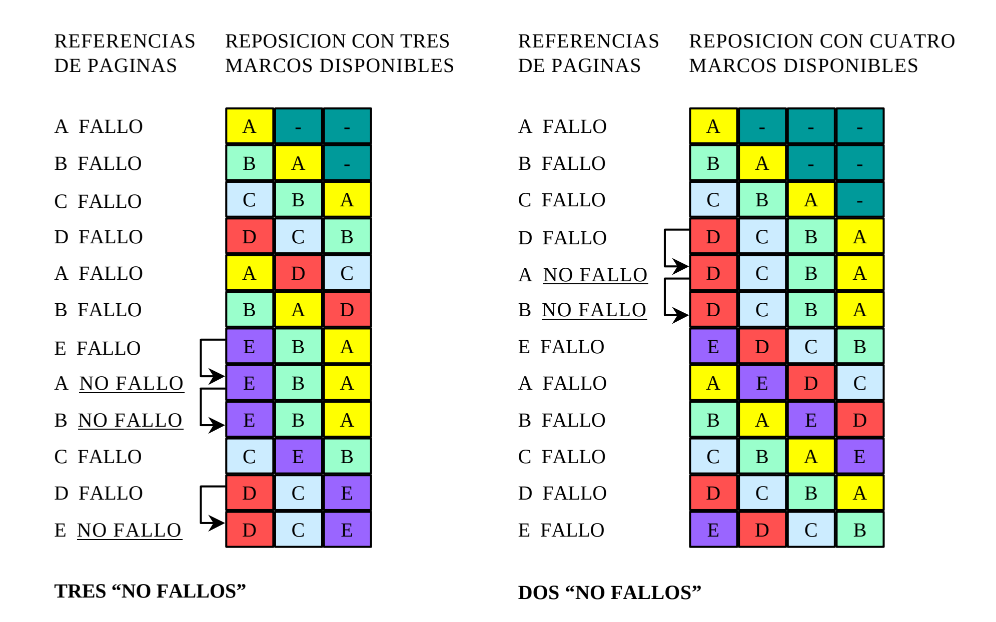


Figura 3.34: Ejemplo de anomalía FIFO.

- Ej.: La página de uso menos frecuente puede ser la página de entrada más reciente al almacenamiento principal, y por lo tanto existe una alta probabilidad de que sea usada de inmediato.

3.15.7 Reposición de Página No Usada - Recientemente (NUR)

Presupone que las páginas que no han tenido uso reciente tienen poca probabilidad de ser usadas en el futuro próximo y pueden ser reemplazadas por otras nuevas.

Es deseable reemplazar una página que no ha sido cambiada mientras estaba en el almacenamiento primario.

La estrategia NUR se implementa con la adición de *dos bits de hardware por página*:

- "*Bit referenciado*":
  - = 0 si la página no ha sido referenciada.
  - = 1 si la página ha sido referenciada.
- "*Bit modificado*" (también llamado "*bit sucio*"):
  - = 0 si la página no ha sido modificada.
  - = 1 si la página ha sido modificada.

La selección de la página que será reemplazada comienza buscando una página que no ha sido referenciada, pero si no la encuentra habrá que reemplazar una página que ha sido referenciada.

Si una página ha sido referenciada se comprueba si ha sido modificada o no:

- Si no ha sido modificada se la reemplaza:
  - Su reposición representa menos sobrecarga que la de una página modificada, ya que debería grabarse de nuevo en el almacenamientos secundario.
- Si no se encuentra una página que no ha sido modificada será reemplazada una página modificada.

Con el transcurso del tiempo la mayoría de los "*bits referenciados*" serán activados:

- Se pierde la capacidad para distinguir las páginas más deseables para ser reemplazadas.
- Para evitarlo se ajustan periódicamente todos los "*bits referenciados*" a "0":
  - Se logra un nuevo inicio.
  - Se vuelve vulnerable al reemplazo aún a las páginas activas, pero solo brevemente, mientras se reajustan los bits.

Los "*bits modificados*" **no** se ajustan periódicamente según esta estrategia.

3.16 Localidad

El concepto de "*localidad*" expresa [7, Deitel]:

- "*Los procesos tienden a hacer referencia al almacenamiento en patrones no uniformes y muy localizados*".

La "*localidad*" se manifiesta en el "*tiempo*" y en el "*espacio*":

- Es una propiedad empírica (observada).
- Nunca está garantizada pero es altamente probable.
- Ej.: Los procesos tienden a favorecer ciertos subconjuntos de páginas, las que tienden a ser adyacentes entre sí en el espacio de direcciones virtuales del proceso.
- Está relacionada con la forma en que se escriben los programas y se organizan los datos.

"*Localidad temporal*": significa que las localidades de almacenamiento referenciadas recientemente tienen una alta probabilidad de ser referenciadas en un futuro próximo:

- Se apoya en la utilización de:
  - Formación de ciclos (loops).
  - Subrutinas.
  - Pilas.

- Variables usadas para contar y totalizar.

"*Localidad en el espacio*": significa que las referencias de almacenamiento tienden a acumularse de manera tal que, una vez que se hace referencia a una localidad, es muy probable que las localidades cercanas sean también referenciadas:

- Se apoya en la utilización de:
  - Recorrido de arreglos.
  - Ejecución secuencial de código.
  - Tendencia de los programadores a colocar definiciones de variables relacionadas, próximas entre sí.

Un programa puede ejecutar eficientemente mientras su subconjunto de páginas preferido se encuentre en el almacenamiento primario.

El número de fallos de páginas de un proceso depende de la cantidad de almacenamiento primario disponible para sus páginas.

Generalmente los procesos no muestran patrones de referencias aleatorios uniformemente distribuidos por sus diferentes páginas.

Al reducir el número de marcos (celdas) de páginas disponibles para un proceso existe un intervalo durante el cual la razón de fallos de páginas no se afecta excesivamente.

En determinado punto, cuando se reduce más el número de marcos de páginas, el número de fallos de páginas aumenta drásticamente.

Mientras el subconjunto de páginas favorecidas por un proceso permanezca en el almacenamiento primario, el número de fallos de páginas no aumenta mucho.

Tan pronto como las páginas del *subconjunto favorecido* son retiradas del almacenamiento primario, la actividad de paginación del proceso aumenta en gran medida al referenciar y traer de nuevo estas páginas al almacenamiento primario.

Los "*subconjuntos favorecidos*" también son llamados "*conjuntos de trabajo*" o "*working sets*".[31]

3.17 Conjuntos de Trabajo

Denning desarrolló un punto de vista de la actividad de paginación de un programa llamado la "*teoría de conjunto de trabajo del comportamiento de un programa*" [7, Deitel].

Un "*conjunto de trabajo*" es una *colección de páginas a las cuales un proceso hace activamente referencia.*

Denning sostenía que para que un programa se ejecutara eficientemente, su conjunto de trabajo debe ser mantenido en el almacenamiento primario, para evitar la "*hiperpaginación*".

Una "*política de administración de almacenamiento por conjunto de trabajo*" trata de mantener el conjunto de trabajo de los programas activos en el almacenamiento primario.

La decisión de añadir un nuevo proceso al conjunto activo de procesos (aumentar el nivel de multiprogramación):

[31] Ver Figura 3.35 de la página 110 [7, Deitel].

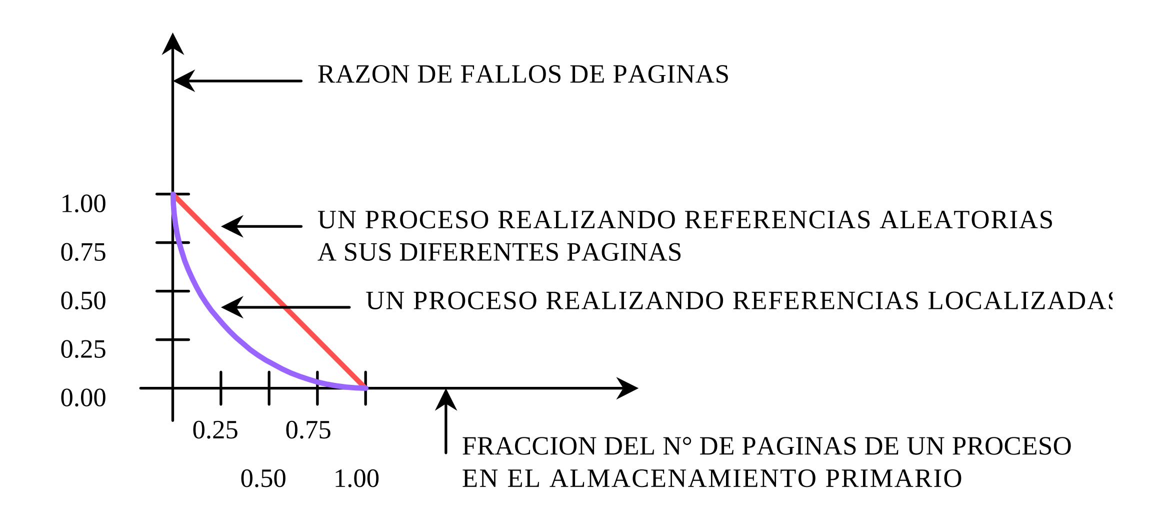


Figura 3.35: Fenómeno de localidad.

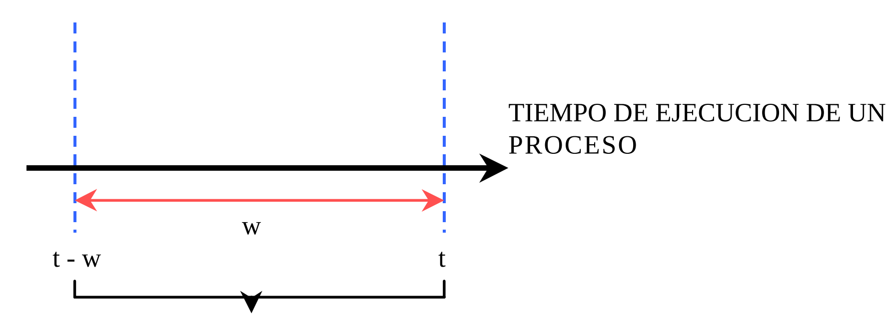


Figura 3.36: Una definición del conjunto de trabajo de páginas de un proceso.

- Se basa en si hay suficiente espacio disponible en el almacenamiento primario como para acomodar el conjunto de trabajo del nuevo proceso.
- Se toma generalmente de forma heurística ya que es *imposible* para el sistema conocer por anticipado el tamaño del conjunto de trabajo de un proceso dado.

El conjunto de trabajo de páginas de un proceso "*w(t,w)*" en el momento "*t*" es el conjunto de páginas referidas por un proceso durante el intervalo de tiempo del proceso "*t - w*" a "*t*".[32]

El "*tiempo del proceso*" es el tiempo durante el cual este proceso tiene la cpu.

La variable "*w*" se denomina "*tamaño de la ventana del conjunto de trabajo*":

- La determinación del tamaño de "*w*" es muy importante.

[32] Ver Figura 3.36 de la página 110 [7, Deitel].

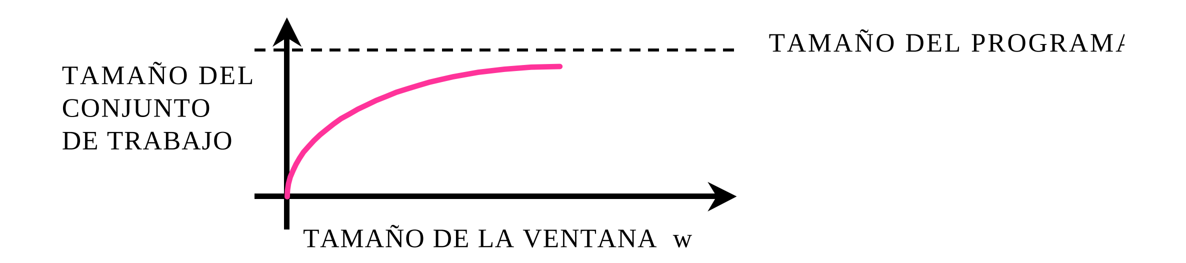


Figura 3.37: Tamaño del conjunto de trabajo como una función del tamaño de la ventana.

- Al aumentar el tamaño de la ventana "*w*" aumenta el tamaño del conjunto de trabajo.[33]

"*El verdadero conjunto de trabajo de un proceso es el conjunto de páginas que deben estar en el almacenamiento primario para la ejecución eficaz de este proceso*".

Los conjuntos de trabajo **cambian** mientras un proceso está en ejecución:

- Complica la administración precisa del almacenamiento primario en base a esta estrategia.
- "*Los conjuntos de trabajo son transitorios y el siguiente conjunto de trabajo del proceso puede diferir substancialmente de su conjunto de trabajo anterior*".
- Se debe evitar un exceso de compromiso del almacenamiento primario y la consecuente hiperpaginación.

3.18 Paginación por Demanda y Paginación Anticipada

3.18.1 Paginación por Demanda

Las paginas son cargadas por demanda [7, Deitel].

No se llevan páginas del almacenamiento secundario al primario hasta que son referenciadas explícitamente por un proceso en ejecución.

Las razones del *atractivo* de esta estrategia son:

- Los resultados de computabilidad, en especial el "*problema de parada*", indican que *el camino que tomará la ejecución de un programa no se puede predecir con exactitud.*
- Garantiza que solo las páginas que necesita el proceso sean traídas al almacenamiento principal.
- La sobrecarga de proceso para decidir qué página traer al almacenamiento principal es mínima.

[33] Ver Figura 3.37 de la página 111 [7, Deitel].

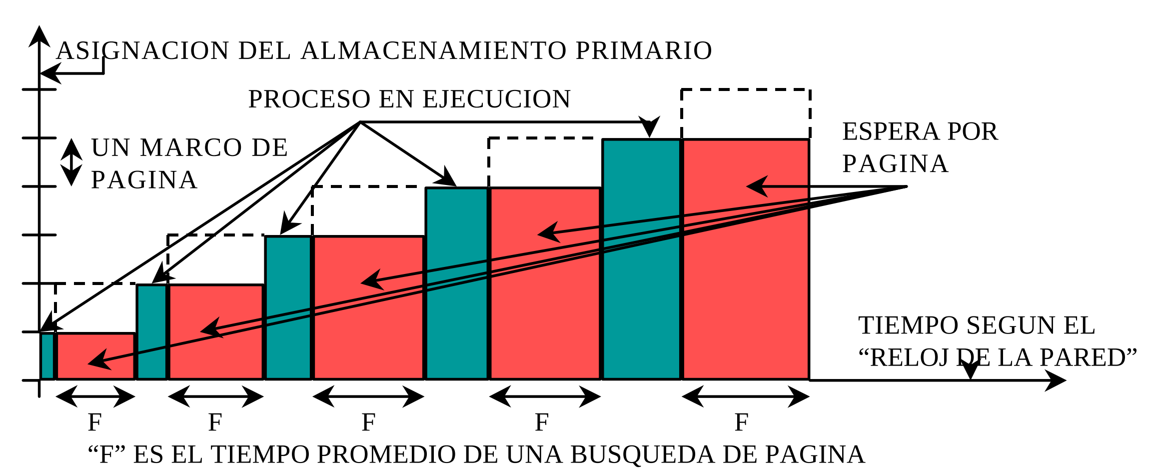


Figura 3.38: Producto espacio - tiempo con paginación por demanda.

El principal inconveniente está en los procesos que requieren acumular sus páginas una por una:

- Los tiempos de espera de páginas son considerables.
- Es creciente la cantidad de almacenamiento primario afectada al proceso que espera páginas, por lo que el "*producto espacio - tiempo*" se incrementa.

El "*producto espacio - tiempo*" indica la cantidad de almacenamiento que usa un proceso y la cantidad de tiempo que lo usa.

"*La reducción del producto espacio - tiempo de las esperas de páginas de un proceso es una meta importante de las estrategias de administración del almacenamiento*".[34]

3.18.2 Paginación Anticipada

El S. O. intenta predecir las páginas que un proceso va a necesitar y a continuación precarga estas páginas cuando hay espacio disponible [7, Deitel].

Mientras el proceso ejecuta sus páginas actuales, el sistema carga páginas nuevas que estarán disponibles cuando el proceso las pida, debido a ello, el tiempo de ejecución de un proceso se puede reducir.

3.19 Liberación de Página y Tamaño de Página

3.19.1 Liberación de Página

Un proceso usuario puede emitir una "*liberación voluntaria de página*" para liberar el marco de página cuando ya no necesitara esa página [7, Deitel].

Se puede eliminar el "*desperdicio*" y acelerar la ejecución.

[34] Ver Figura 3.38 de la página 112 [7, Deitel].

El inconveniente es que la incorporación de mandatos de liberación de páginas dentro de los programas de usuarios puede ser peligroso y retrasar el desarrollo de aplicaciones.

"*Los compiladores y S. O. deberían detectar automáticamente situaciones de liberación de página mucho antes de lo que es posible con estrategias de conjuntos de trabajo*".

3.19.2 Tamaño de Página

Generalmente el almacenamiento real se divide en marcos o celdas de página de tamaño fijo [7, Deitel].

Los interrogantes tienen que ver con el tamaño de las páginas, si todas las páginas tendrán igual tamaño, si en caso de utilizar páginas de diferente tamaño las páginas mayores deben ser o no múltiplos enteros de las menores, etc.

Algunas consideraciones para determinar el tamaño de página son las siguientes:

- Cuanto más pequeño sea el tamaño de una página, más páginas y marcos de páginas habrá y mayores serán las tablas de páginas:
  - El desperdicio de almacenamiento debido al tamaño excesivo de las tablas de página se llama "*fragmentación de tablas*".
  - Esto indica la necesidad de *páginas más grandes*.
- Con páginas grandes, grandes cantidades de información que nunca llegaría a ser referenciada, se paginarán hacia el almacenamiento primario:
  - Esto indica la necesidad de *páginas más pequeñas*.
- Debido a que las transferencias de e / s del disco (paginación) consumen bastante tiempo, se debe minimizar la paginación que un proceso requiera:
  - Esto indica la necesidad de *páginas grandes*.
- Los programas tienden a mostrar la propiedad de *localidad de referencia* y esta localidad tiende a ser pequeña:
  - Esto indica la necesidad de *páginas pequeñas*.
- Los procedimientos y datos rara vez comprenden un número entero de páginas, por lo que los sistemas de paginación experimentan una "*fragmentación interna*":
  - El desperdicio promedio es de 1 / 2 página no usada por segmento (grupo) de páginas, que estará en la última página del segmento.
  - Esto indica la necesidad de *páginas pequeñas*.

Los tamaños de pagina mas utilizados son: 512 b, 1 kb, 2 kb, 4 kb.

3.20 Comportamiento de un Programa en la Paginación

Respecto del porcentaje de las páginas de un proceso típico referenciadas desde el momento de iniciarse su ejecución [7, Deitel]:

- Un proceso tiende a hacer referencia a una parte significativa de sus páginas inmediatamente después de iniciar su ejecución.
- El proceso puede concluir sin haber referenciado a algunas de sus páginas, correspondientes a rutinas que atienden errores que no se produjeron.

Respecto de variar el tamaño de la página manteniendo constante la cantidad de almacenamiento primario:

- El número de fallos de páginas experimentados por un proceso en ejecución tiende a aumentar con el tamaño de la página, debido a que se traen al almacenamiento primario un mayor número de procedimientos y datos que no serán referenciados, restando lugar para los que sí lo serán.

Respecto de cómo el *promedio de tiempo interfallos* (*tiempo entre fallos de página*) varía al aumentar el número de marcos de página asignados al proceso:

- Cuanto más marcos de página tenga un proceso, mayor será el tiempo entre los fallos de páginas.
- El punto de inflexión se da cuando el proceso tiene todo su conjunto de trabajo en el almacenamiento primario.
- Asignar marcos de página adicionales más allá del punto de inflexión no produce efectos significativos sobre el tiempo interfallos.

Respecto del porcentaje de instrucciones de una página que son ejecutadas antes de transferirse el control a otra página, los valores experimentales obtenidos indican un máximo de 200 instrucciones por página de 1 kb.[35]

[35] Ver Figura 3.39 de la página 115 [7, Deitel].

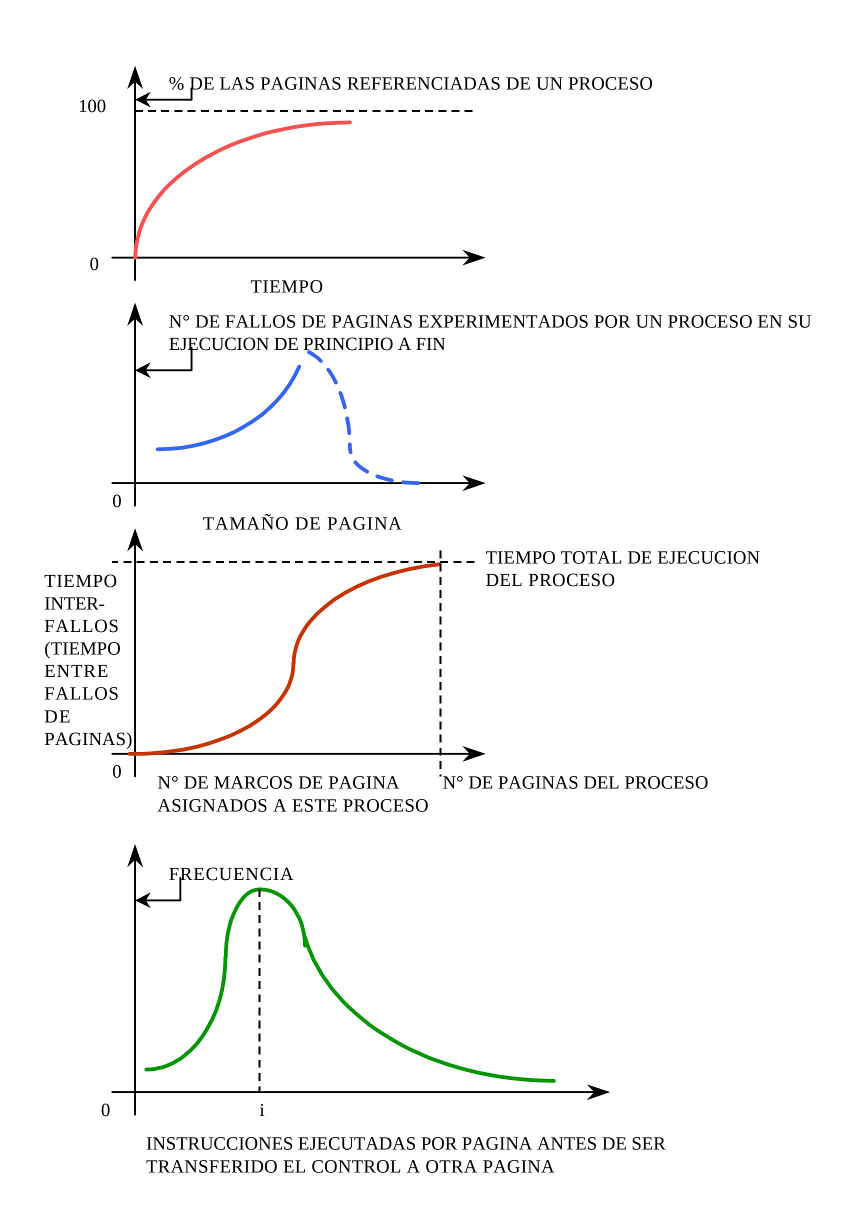


Figura 3.39: Comportamiento de un programa en la paginación.

Capítulo 4

Sistemas de Archivos

4.1 Introducción

Todas las aplicaciones computarizadas necesitan almacenar y recuperar la información [7, Deitel]:

- Superando las limitaciones del almacenamiento real.
- Trascendiendo a la duración de los procesos que las utilizan o generan.
- Independizando a la información de los procesos permitiendo el acceso a la misma a través de varios procesos.

Las condiciones esenciales para el almacenamiento de la información a largo plazo son:

- Debe ser posible almacenar una cantidad muy grande de información.
- La información debe sobrevivir a la conclusión del proceso que la utiliza.
- Debe ser posible que varios procesos tengan **acceso concurrente** a la información.

La solución es el almacenamiento de la información en discos y otros medios externos en unidades llamadas **archivos**:

- Los archivos deben ser **persistentes**, decir que no deben verse afectados por la creación o terminación de un proceso.
- Los archivos son una colección de datos con nombre.
- Pueden ser manipulados como una unidad por operaciones como: open, close, create, destroy, copy, rename, list.
- Los elementos de datos individuales dentro del archivo pueden ser manipulados por operaciones como: read, write, update, insert, delete.

El "Sistema de Archivos" es la parte del sistema de administración del almacenamiento responsable, principalmente, de la administración de los archivos del almacenamiento secundario.

Es la parte del S. O. responsable de permitir "*compartir controladamente*" la información de los archivos.

4.2 Funciones del Sistema de Archivos

Los usuarios deben poder crear, modificar y borrar archivos.

Se deben poder compartir los archivos de una manera cuidadosamente controlada [7, Deitel].

El mecanismo encargado de compartir los archivos debe proporcionar varios tipos de acceso controlado:

- Ej.: "*Acceso de Lectura*", "*Acceso de Escritura*", "*Acceso de Ejecución*", varias combinaciones de estos, etc.

Se debe poder estructurar los archivos de la manera más apropiada a cada aplicación.

Los usuarios deben poder ordenar la transferencia de información entre archivos.

Se deben proporcionar posibilidades de "*respaldo*" y "*recuperación*" para prevenirse contra:

- La pérdida accidental de información.
- La destrucción maliciosa de información.

Se debe poder referenciar a los archivos mediante "*Nombres Simbólicos*", brindando "*Independencia de Dispositivos*".

En ambientes sensibles, el sistema de archivos debe proporcionar posibilidades de "*Cifrado*" y "*Descifrado*".

El sistema de archivos debe brindar una interfase favorable al usuario:

- Debe suministrar una "*visión lógica*" de los datos y de las funciones que serán ejecutadas, en vez de una "*visión física*".
- El usuario no debe tener que preocuparse por:
  - Los dispositivos particulares.
  - Dónde serán almacenados los datos.
  - El formato de los datos en los dispositivos.
  - Los medios físicos de la transferencia de datos hacia y desde los dispositivos.

4.3 El Sistema de Archivos

Un "*Archivo*" es un conjunto de registros relacionados [23, Tanenbaum].

El "*Sistema de Archivos*" es un componente importante de un S. O. y suele contener [7, Deitel]:

- "*Métodos de acceso*" relacionados con la manera de acceder a los datos almacenados en archivos.
- "*Administración de archivos*" referida a la provisión de mecanismos para que los archivos sean almacenados, referenciados, compartidos y asegurados.

- "*Administración del almacenamiento auxiliar*" para la asignación de espacio a los archivos en los dispositivos de almacenamiento secundario.
- "*Integridad del archivo*" para garantizar la integridad de la información del archivo.

El sistema de archivos está relacionado especialmente con la administración del espacio de almacenamiento secundario, fundamentalmente con el almacenamiento de disco.

Una forma de organización de un sistema de archivos puede ser la siguiente:

- Se utiliza una "*raíz*" para indicar en qué parte del disco comienza el "*directorio raíz*".
- El "*directorio raíz*" apunta a los "*directorios de usuarios*".
- Un "*directorio de usuario*" contiene una entrada para cada uno de los archivos del usuario.
- Cada entrada de archivo apunta al lugar del disco donde está almacenado el archivo referenciado.

Los nombres de archivos solo necesitan ser únicos dentro de un directorio de usuario dado.

El nombre del sistema para un archivo dado debe ser único para el sistema de archivos.

En sistemas de archivo "*jerárquicos*" el nombre del sistema para un archivo suele estar formado como el "*nombre de la trayectoria*" del directorio raíz al archivo.

4.4 Archivos

Se considerará el punto de vista del usuario.

4.4.1 Nombre de los Archivos

Las reglas exactas para los nombres de archivos varían de sistema a sistema [23, Tanenbaum].

Algunos sistemas de archivos distinguen entre las letras mayúsculas y minúsculas, mientras que otros no.

Muchos S. O. utilizan nombres de archivo con dos partes, separadas por un punto:

- La parte posterior al punto es la *extensión de archivo* y generalmente indica algo relativo al archivo, aunque las extensiones suelen ser meras convenciones.

4.4.2 Estructura de un Archivo

Los archivos se pueden estructurar de varias maneras, las más comunes son [23, Tanenbaum]:

- "*Secuencia de bytes*":

  - El archivo es una serie no estructurada de bytes.
  - Posee máxima flexibilidad.
  - El S. O. no ayuda pero tampoco estorba.

- "*Secuencia de registros*":

  - El archivo es una secuencia de registros de longitud fija, cada uno con su propia estructura interna.

- "*Arbol*":

  - El archivo consta de un árbol de registros, no necesariamente de la misma longitud.
  - Cada registro tiene un **campo key** (**llave** o **clave**) en una posición fija del registro.
  - El árbol se ordena mediante el campo de clave para permitir una rápida búsqueda de una clave particular.

4.4.3 Tipos de Archivos

Muchos S. O. soportan varios tipos de archivos, por ej.: *archivos regulares, directorios, archivos especiales de caracteres, archivos especiales de bloques*, etc., donde [23, Tanenbaum]:

- Los **Archivos Regulares** son aquellos que contienen información del usuario.
- Los **Directorios** son archivos de sistema para el mantenimiento de una estructura del sistema de archivos.
- Los **Archivos Especiales de Caracteres:**

  - Tienen relación con la e / s.
  - Se utilizan para modelar dispositivos seriales de e / s (terminales, impresoras, redes, etc.).

- Los **Archivos Especiales de Bloques** se utilizan para modelar discos.

4.4.4 Acceso a un Archivo

Los *tipos de acceso* más conocidos son:

- **Acceso Secuencial:** el proceso lee en orden todos los registros del archivo comenzando por el principio, sin poder:

  - Saltar registros.
  - Leer en otro orden.

- **Acceso Aleatorio:** el proceso puede leer los registros en cualquier orden utilizando dos métodos para determinar el punto de inicio de la lectura:
  - Cada operación de lectura (read) da la posición en el archivo con la cual iniciar.
  - Una operación especial (seek) establece la posición de trabajo pudiendo luego leerse el archivo secuencialmente.

4.4.5 Atributos de Archivo

Cada archivo tiene:

- Su nombre y datos.
- Elementos adicionales llamados **atributos**, que varían considerablemente de sistema a sistema.

Algunos de los **posibles atributos de archivo** son [23, Tanenbaum]:

- "Protección": quién debe tener acceso y de qué forma.
- "Contraseña": contraseña necesaria para acceder al archivo.
- "Creador": identificador de la persona que creó el archivo.
- "Propietario": propietario actual.
- "Bandera exclusivo - para - lectura": 0 lectura / escritura, 1 para lectura exclusivamente.
- "Bandera de ocultamiento": 0 normal, 1 para no exhibirse en listas.
- "Bandera de sistema": 0 archivo normal, 1 archivo de sistema.
- "Bandera de biblioteca": 0 ya se ha respaldado, 1 necesita respaldo.
- "Bandera ascii / binario": 0 archivo en ascii, 1 archivo en binario.
- "Bandera de acceso aleatorio": 0 solo acceso secuencial, 1 acceso aleatorio.
- "Bandera temporal": 0 normal, 1 eliminar al salir del proceso.
- "Banderas de cerradura": 0 no bloqueado, distinto de 0 bloqueado.
- "Longitud del registro": número de bytes en un registro.
- "Posición de la llave": ajuste de la llave dentro de cada registro.
- "Longitud de la llave": número de bytes en el campo llave.
- "Tiempo de creación": fecha y hora de creación del archivo.
- "Tiempo del último acceso": fecha y hora del último acceso al archivo.
- "Tiempo de la última modificación": fecha y hora de la última modificación al archivo.
- "Tamaño actual": número de bytes en el archivo.
- "Tamaño máximo": tamaño máximo al que puede crecer el archivo.

4.4.6 Operaciones con Archivos

Las llamadas más comunes al sistema relacionadas con los archivos son [23, Tanenbaum]:

- Create (crear): el archivo se crea sin datos.
- Delete (eliminar): si el archivo ya no es necesario debe eliminarse para liberar espacio en disco. Ciertos S. O. eliminan automáticamente un archivo no utilizado durante "n" días.
- Open (abrir): antes de utilizar un archivo, un proceso debe abrirlo. La finalidad es permitir que el sistema traslade los atributos y la lista de direcciones en disco a la memoria principal para un rápido acceso en llamadas posteriores.
- Close (cerrar): cuando concluyen los accesos, los atributos y direcciones del disco ya no son necesarios, por lo que el archivo debe cerrarse y liberar la tabla de espacio interno.
- Read (leer): los datos se leen del archivo; quien hace la llamada debe especificar la cantidad de datos necesarios y proporcionar un buffer para colocarlos.
- Write (escribir): los datos se escriben en el archivo, en la posición actual. El tamaño del archivo puede aumentar (agregado de registros) o no (actualización de registros).
- Append (añadir): es una forma restringida de "write". Solo puede añadir datos al final del archivo.
- Seek (buscar): especifica el punto donde posicionarse. Cambia la posición del apuntador a la posición activa en cierto lugar del archivo.
- Get attributes (obtener atributos): permite a los procesos obtener los atributos del archivo.
- Set attributes (establecer atributos): algunos atributos pueden ser determinados por el usuario y modificados luego de la creación del archivo. La información relativa al modo de protección y la mayoría de las banderas son un ejemplo obvio.
- Rename (cambiar de nombre): permite modificar el nombre de un archivo ya existente.

4.4.7 Archivos Mapeados a Memoria

Algunos S. O. permiten asociar los archivos con un espacio de direcciones de un proceso en ejecución [23, Tanenbaum].

Se utilizan las llamadas al sistema "**map**" y "**unmap**":

- "Map": utiliza un nombre de archivo y una dirección virtual y hace que el S. O. asocie al archivo con la dirección virtual en el espacio de direcciones, por lo cual las lecturas o escrituras de las áreas de memoria asociadas al archivo se efectúan también sobre el archivo mapeado.

- "Unmap": elimina los archivos del espacio de direcciones y concluye la operación de asociación.

El mapeo de archivos elimina la necesidad de programar la e / s directamente, facilitando la programación.

Los principales problemas relacionados son:

- Imposibilidad de conocer a priori la longitud del archivo de salida, el que podría superar a la memoria.
- Dificultad para compartir los archivos mapeados evitando inconsistencias, ya que las modificaciones hechas en las páginas no se verán reflejadas en el disco hasta que dichas páginas sean eliminadas de la memoria.

4.5 Directorios

Generalmente son utilizados por los S. O. para llevar un registro de los archivos [23, Tanenbaum].

En muchos sistemas son a su vez también archivos.

4.5.1 Sistemas Jerárquicos de Directorios

El directorio contiene un conjunto de datos por cada archivo referenciado.

Una posibilidad es que el directorio contenga por cada archivo referenciado [7, Deitel]:

- El nombre.
- Sus atributos.
- Las direcciones en disco donde se almacenan los datos.

Otra posibilidad es que cada entrada del directorio contenga:

- El nombre del archivo.
- Un apuntador a otra estructura de datos donde se encuentran los atributos y las direcciones en disco.

Al *abrir un archivo* el S. O.:

- Busca en su directorio el nombre del archivo.
- Extrae los atributos y direcciones en disco.
- Graba esta información en una tabla de memoria real.
- Todas las referencias subsecuentes al archivo utilizarán la información de la memoria principal.

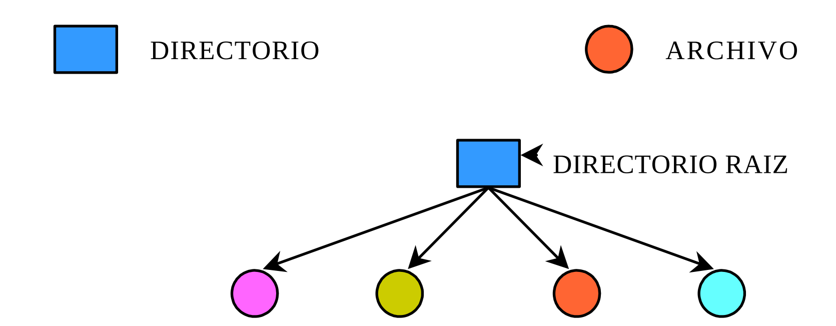


Figura 4.1: Un solo directorio compartido por todos los usuarios.

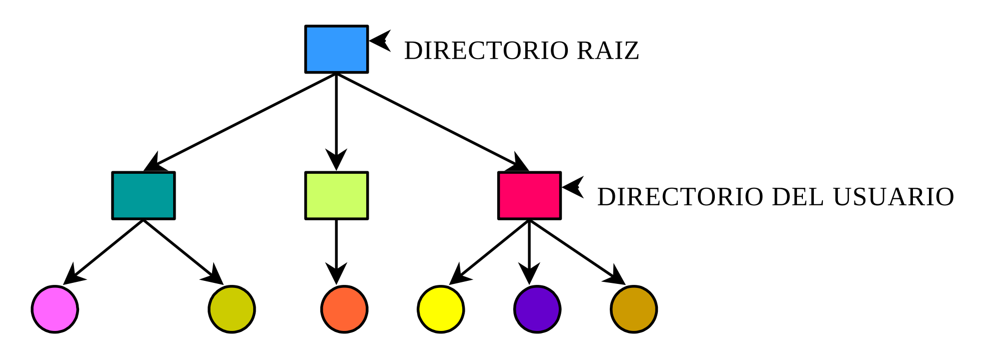


Figura 4.2: Un directorio por usuario.

El *número y organización de directorios* varía de sistema en sistema:

- Directorio único: el sistema tiene un solo directorio con todos los archivos de todos los usuarios.[1]

- Un directorio por usuario: el sistema habilita un solo directorio por cada usuario.[2]

- Un árbol de directorios por usuario: el sistema permite que cada usuario tenga tantos directorios como necesite, respetando una jerarquía general.[3]

4.5.2 Nombre de las Rutas de Acceso

Cuando el sistema de archivos está organizado como un árbol de directorios se necesita una forma de determinar los nombres de los archivos.

Los **principales métodos** para nombres de los archivos son [23, Tanenbaum]:

[1] Ver Figura 4.1 de la página 124 [23, Tanenbaum].
[2] Ver Figura 4.2 de la página 124 [23, Tanenbaum].
[3] Ver Figura 4.3 de la página 125 [23, Tanenbaum].

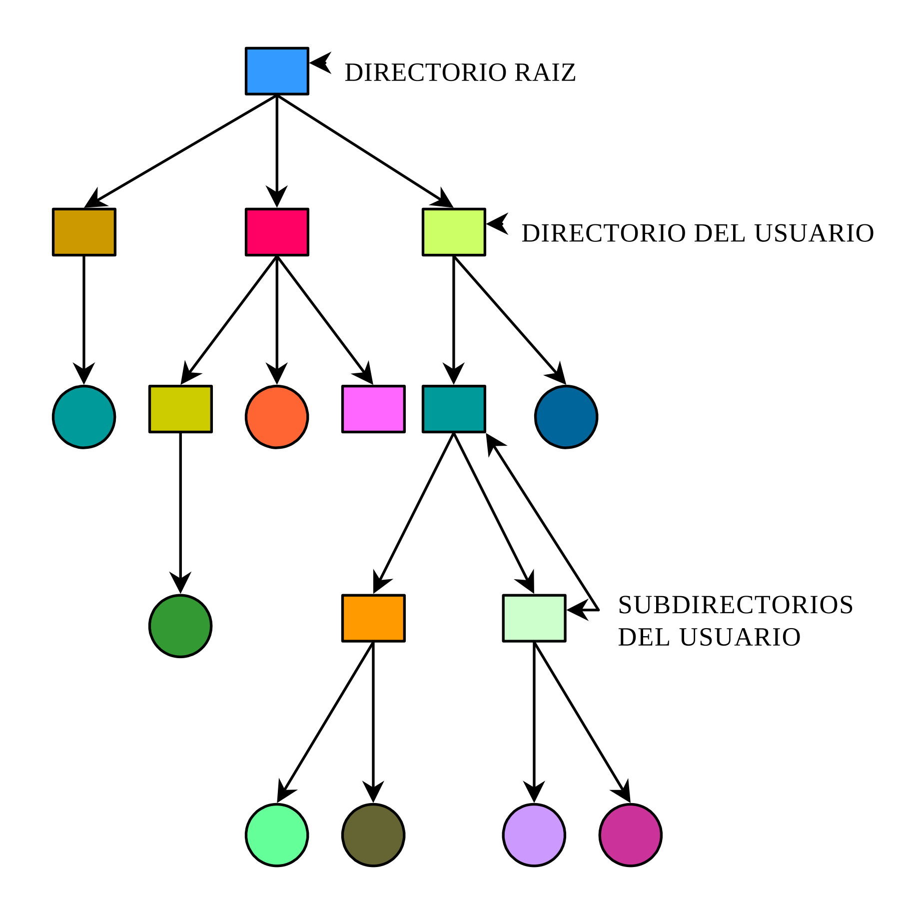


Figura 4.3: Un árbol arbitrario por usuario.

- **Ruta de Acceso Absoluta:**
  - Cada archivo tiene una ruta de acceso absoluta.
  - Consta de la ruta de acceso desde el directorio raíz hasta el archivo.
  - Los componentes de la ruta de acceso se separan mediante algún caracter llamado *"separador"*.
- **Ruta de Acceso Relativa:**
  - Se utiliza junto con el concepto de directorio de trabajo o directorio activo.
  - Todos los nombres que no comiencen en el directorio raíz se toman en relación con el directorio de trabajo.
  - El nombre absoluto de la ruta de acceso siempre funciona, sin importar cual sea el directorio de trabajo.

4.5.3 Operaciones con Directorios

Las llamadas al sistema permitidas para el manejo de los directorios tienen variación de sistema a sistema [23, Tanenbaum].

Las más comunes son las siguientes:

- Create (crear): se crea un directorio vacío.
- Delete (eliminar): se elimina un directorio, que debe estar vacío.
- Opendir (abrir directorio): se pueden leer los directorios:
  - Antes de poder leer un directorio, éste debe ser abierto.
- Closedir (cerrar directorio): cuando se ha leído un directorio, éste debe ser cerrado para liberar el espacio correspondiente de la tabla interna.
- Readdir (leer directorio): regresa la siguiente entrada en un directorio abierto, sin importar el tipo de estructura de directorios que se utilice.
- Rename (cambiar de nombre): cambia el nombre de un directorio de manera similar al cambio para archivos.
- Link (ligar): es una técnica que permite que un archivo aparezca en más de un directorio:
  - Especifica un archivo existente y el nombre de una ruta de acceso.
  - Crea un enlace del archivo ya existente con el nombre especificado en la ruta de acceso.
- Unlink (desligar): se elimina una entrada del directorio:

- Si el archivo que se desea desligar aparece solo en un directorio (el caso normal):
  * Se elimina del sistema de archivos.
- Si el archivo que se desea desligar, está presente en varios directorios:
  * Solo se elimina la ruta de acceso especificada.
  * Las demás rutas permanecen.

4.6 Implantación del Sistema de Archivos y sus Relaciones con la Asignación y Liberación de Espacio

Se consideran **aspectos** tales como [7, Deitel]:

- La *forma de almacenamiento de archivos y directorios.*
- La *administración del espacio en disco.*
- La *forma de hacerlo de manera eficiente y confiable.*

Se deben tener presentes **problemas** tales como la "*fragmentación*" creciente del espacio en disco:

- Ocasiona problemas de performance al hacer que los archivos se desperdiguen a través de bloques muy dispersos.
- Una técnica para aliviar el problema de la "*fragmentación*" consiste en realizar periódicamente:
  - "*Condensación*": se pueden "*reorganizar*" los archivos expresamente o automáticamente según algún criterio predefinido.
  - "*Recolección de basura o residuos*": se puede hacer fuera de línea o en línea, con el sistema activo, según la implementación.

4.6.1 Implantación de Archivos

El aspecto clave de la implantación del almacenamiento de archivos es el *registro de los bloques asociados a cada archivo* [7, Deitel].

Algunos de los métodos utilizados son los siguientes:

- *Asignación contigua o adyacente:*
  - Los archivos son asignados a áreas contiguas de almacenamiento secundario.
  - Las principales **ventajas** son:
    * Facilidad de implantación, ya que solo se precisa el número del bloque de inicio para localizar un archivo.
    * Rendimiento excelente respecto de la e / s.
  - Los principales **defectos** son:

    * Se debe conocer el tamaño máximo del archivo al crearlo.
    * Produce una gran fragmentación de los discos.

- *Asignación no contigua:*
  - Son esquemas de almacenamiento más dinámicos, destacándose los siguientes:
  - *Asignación encadenada orientada hacia el sector:*
    * El disco se considera compuesto de sectores individuales.
    * Los archivos constan de varios sectores que pueden estar dispersos por todo el disco.
    * Los sectores que pertenecen a un archivo común contienen apuntadores de uno a otro formando una "*lista encadenada*".
    * Una "*lista de espacio libre*" contiene entradas para todos los sectores libres del disco.
    * Las ampliaciones o reducciones en el tamaño de los archivos se resuelven actualizando la "*lista de espacio libre*" y no hay necesidad de condensación.
    * Las principales **desventajas** son:
      · Debido a la posible dispersión en el disco, la recuperación de registros lógicamente contiguos puede significar largas búsquedas.
      · El mantenimiento de la estructura de "*listas encadenadas*" significa una sobrecarga en tiempo de ejecución.
      · Los apuntadores de la estructura de lista consumen espacio en disco.
  - *Asignación por bloques:*
    * Es más eficiente y reduce la sobrecarga en ejecución.
    * Es una mezcla de los métodos de asignación contigua y no contigua.
    * Se asignan bloques de sectores contiguos en vez de sectores individuales.
    * El sistema trata de asignar nuevos bloques a un archivo eligiendo bloques libres lo más próximos posible a los bloques del archivo existentes.
    * Las formas más comunes de implementar la asignación por bloques son:
      · Encadenamiento de bloques.
      · Encadenamiento de bloques de índice.
      · Transformación de archivos orientada hacia bloques.
  - *Encadenamiento de bloques o lista ligada:*
    * Las entradas en el directorio de usuarios apuntan al primer bloque de cada archivo.
    * Cada uno de los bloques de longitud fija que forman un archivo contiene dos partes:
      · Un bloque de datos.
      · Un apuntador al bloque siguiente.
    * Cada bloque contiene varios sectores.
    * Frecuentemente el tamaño de un bloque se corresponde con el de una pista completa del disco.

Figura 4.4: Encadenamiento de bloques o lista ligada de bloques.

- ∗ Localizar un registro determinado requiere:
  - · Buscar en la cadena de bloques hasta encontrar el bloque apropiado.
  - · Buscar en el bloque hasta encontrar el registro.
- ∗ El examen de la cadena desde el principio puede ser lento ya que debe realizarse de bloque en bloque, y pueden estar dispersos por todo el disco.
- ∗ La inserción y el retiro son inmediatos, dado que se deben modificar los apuntadores del bloque precedente.
- ∗ Se pueden usar "*listas de encadenamiento doble*", hacia adelante y hacia atrás, con lo que se facilita la búsqueda.[4]

– *Encadenamiento de bloques de índices:*

- ∗ Los apuntadores son colocados en varios bloques de índices separados:
  - · Cada bloque de índices contiene un número fijo de elementos.
  - · Cada entrada contiene un *identificador de registros* y un *apuntador* a ese registro.

[4] Ver Figura 4.4 de la página 129 [23, Tanenbaum].

        · Si es necesario utilizar más de un bloque de índices para describir un archivo, se encadena una serie de bloques de índices.
    * La gran **ventaja** es que la búsqueda puede realizarse en los propios bloques de índices.
    * Los bloques de índices pueden mantenerse juntos en el almacenamiento secundario para acortar la búsqueda, pero para mejor performance podrían mantenerse en el almacenamiento primario.
    * La principal **desventaja** es que las inserciones pueden requerir la reconstrucción completa de los bloques de índices:
        · Una posibilidad es dejar vacía una parte de los bloques de índices para facilitar inserciones futuras y retardar las reconstrucciones.
    * Es suficiente que el dato del directorio contenga el número de bloque inicial para localizar todos los bloques restantes, sin importar el tamaño del archivo.[5]

- *Transformación de archivos orientada hacia bloques:*
    * Se utilizan números de bloques en vez de apuntadores.
    * Los números de bloques se convierten fácilmente a direcciones de bloques gracias a la geometría del disco.
    * Se conserva un *mapa del archivo*, conteniendo una entrada para cada bloque del disco.
    * Las entradas en el directorio del usuario apuntan a la primera entrada al mapa del archivo para cada archivo.
    * Cada entrada al mapa del archivo contiene el número del bloque siguiente de ese archivo.
    * La entrada al mapa del archivo correspondiente a la última entrada de un archivo determinado se ajusta a algún valor "*centinela*" ("nil") para indicar que se alcanzó el último bloque de un archivo.
    * El sistema puede mantener una *lista de bloques libres*.
    * La principal **ventaja** es que las cercanías físicas del disco se reflejan en el mapa del archivo.[6]

- *Nodos-i (nodos índices):*
    * Se asocia a cada archivo una pequeña **tabla**, llamada nodo-i (nodo índice):
        · Contiene los atributos y direcciones en disco de los bloques del archivo.
        · Se traslada del disco a la memoria principal al abrir el archivo.
        · En rigor, almacena solo las primeras direcciones en disco:
        · o Si el archivo es pequeño, toda la información está en el nodo-i.

[5] Ver Figura 4.5 de la página 131 [7, Deitel].
[6] Ver Figura 4.6 de la página 132 [7, Deitel].

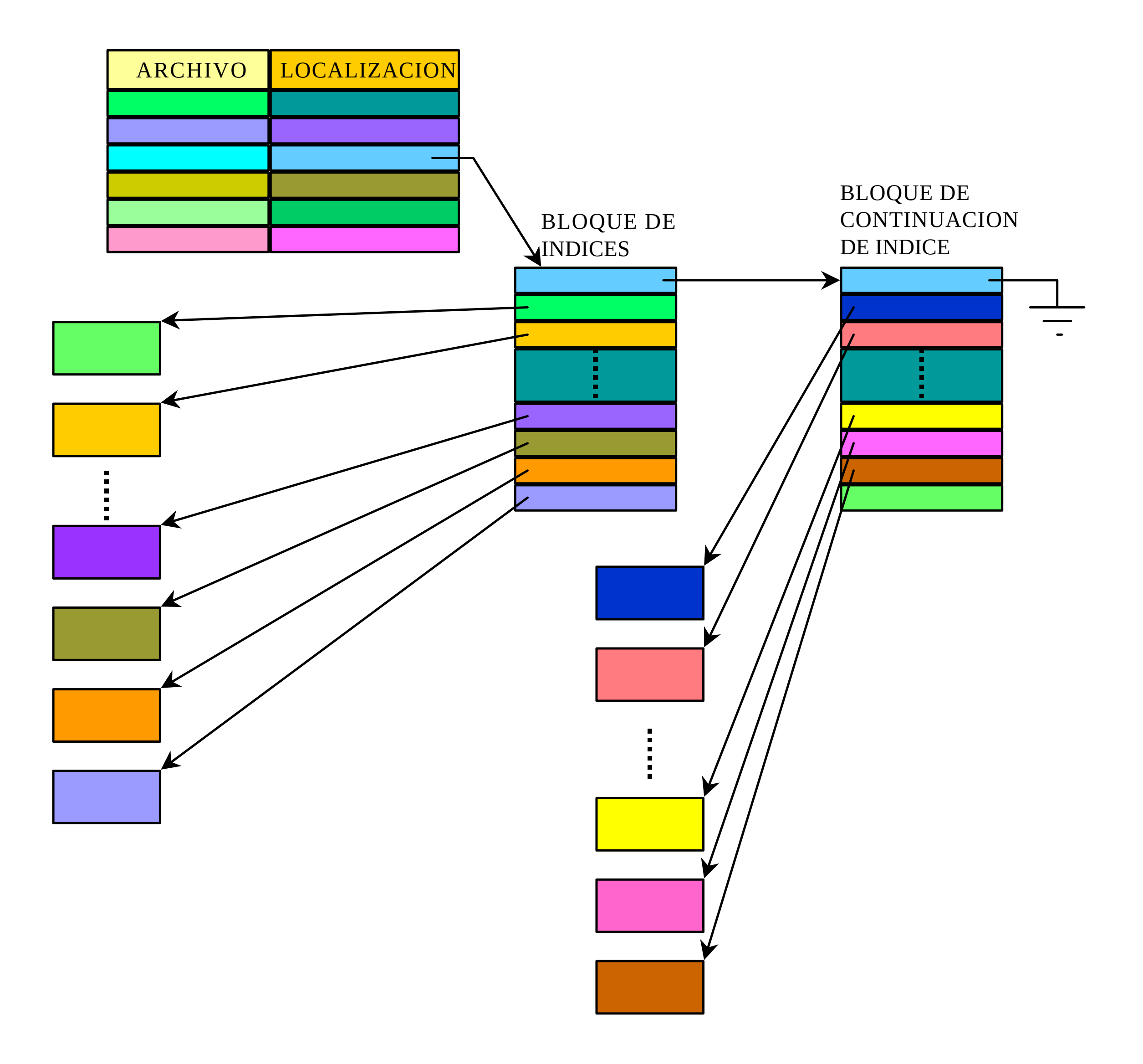


Figura 4.5: Encadenamiento de bloques de índices.

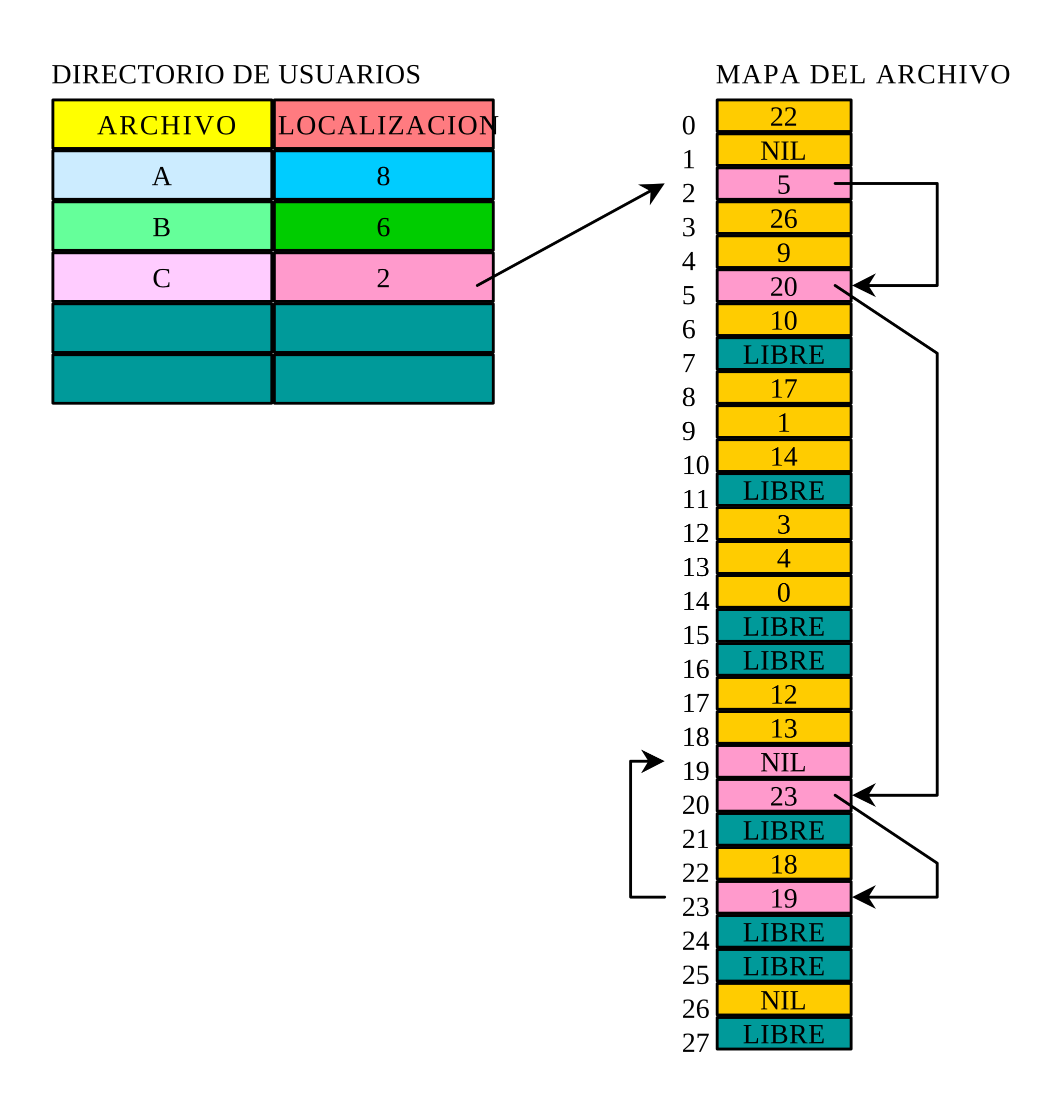


Figura 4.6: Transformación de archivos orientada hacia bloques.

· o Si el archivo es grande, una de las direcciones en el nodo-i es la dirección de un bloque en el disco llamado *bloque simplemente indirecto*:

· - Contiene las direcciones en disco adicionales.

· - Si resulta insuficiente, otra dirección en el nodo-i, el *bloque doblemente indirecto*, contiene la dirección de un bloque que presenta una lista de los bloques simplemente indirectos:

· - Cada bloque simplemente indirecto apunta a un grupo de bloques de datos.

· - De ser necesario se pueden utilizar *bloques triplemente indirectos*.[7]

4.6.2 Implantación de Directorios

Para abrir un archivo el S. O. utiliza información del directorio:

- El directorio contiene la **información** necesaria para encontrar los bloques en el disco.
- El **tipo** de información varía según el sistema.

La principal **función del sistema de directorios** es asociar el nombre del archivo con la información necesaria para localizar los datos.

Un aspecto íntimamente ligado con esto es la posición de almacenamiento de los **atributos**:

- Una posibilidad es almacenarlos en forma directa *dentro del dato del directorio.*
- Otra posibilidad es almacenar los atributos en el *nodo-i* en vez de utilizar la entrada del directorio.

4.6.3 Archivos Compartidos

Frecuentemente conviene que los archivos compartidos aparezcan simultáneamente en distintos directorios de distintos usuarios.

El propio sistema de archivos es una *gráfica dirigida acíclica* en vez de un árbol [23, Tanenbaum].

La conexión entre un directorio y un archivo de otro directorio al cual comparten se denomina **enlace**.

Si los directorios realmente contienen direcciones en disco:

- Se debe tener una copia de las direcciones en disco en el directorio que accede al archivo compartido al enlazar el archivo.
- Se debe evitar que los cambios hechos por un usuario a través de un directorio no sean visibles por los demás usuarios, para lo que se consideraran dos soluciones posibles.

Primer solución:

[7] Ver Figura 4.7 de la página 134 [23, Tanenbaum].

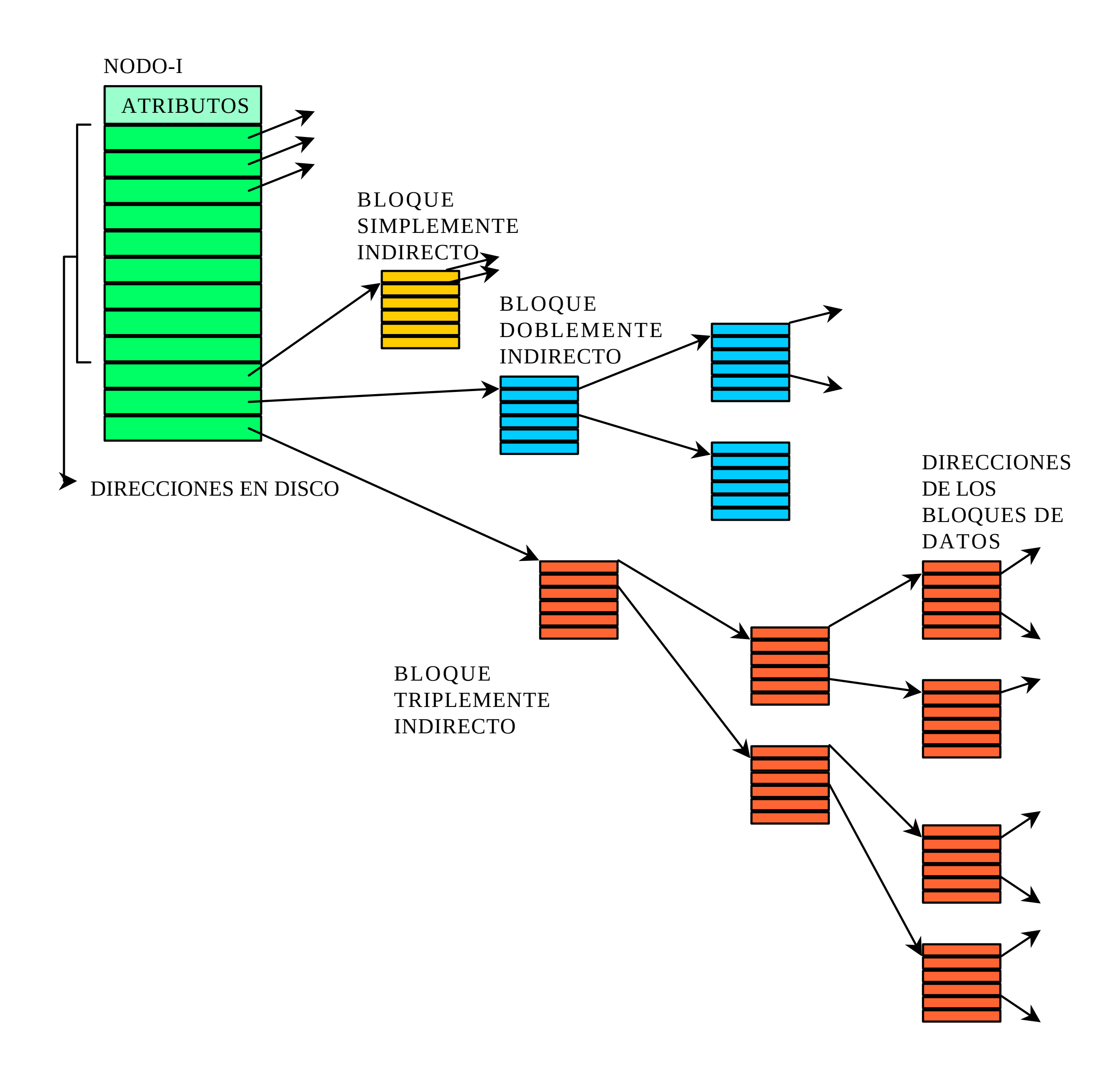


Figura 4.7: Esquema de un nodo-i.

- Los bloques del disco no se enlistan en los directorios, sino en una pequeña estructura de datos asociada al propio archivo.
- Los directorios apuntarían solo a esa pequeña estructura de datos, que podría ser el nodo-i.

Segunda solución:

- El enlace se produce haciendo que el sistema cree un nuevo archivo de tipo "*link*".
- El archivo "*link*":
  - Ingresa al directorio del usuario que accede a un archivo de otro directorio y usuario.
  - Solo contiene el *nombre de la ruta de acceso* del archivo al cual se enlaza.
- Este criterio se denomina **enlace simbólico**.

Desventajas de la primer solución:

- La creación de un enlace:
  - No modifica la propiedad respecto de un archivo.
  - Aumenta el contador de enlaces del nodo-i:
    * El sistema sabe el número de entradas de directorio que apuntan en cierto momento al archivo.
- Si el propietario inicial del archivo intenta eliminarlo, surge un problema para el sistema:
  - Si elimina el archivo y limpia el nodo-i, el directorio que enlazo al archivo tendrá una entrada que apunta a un nodo-i no válido.
  - Si el nodo-i se reasigna a otro archivo el enlace apuntará al archivo incorrecto.
  - El sistema:
    * Puede ver por medio del contador de enlaces en el nodo-i que el archivo sigue utilizándose.
    * No puede localizar todas las entradas de directorio asociadas a ese archivo para eliminarlas.
- La solución podría ser:
  - Eliminar la entrada del directorio inicialmente propietario del archivo.
  - Dejar intacto el nodo-i:
    * Se daría el caso que el directorio que posee el enlace es el único que posee una entrada de directorio para un archivo de otro directorio, para el cual dicho archivo ya no existe.

- * Esto no ocurre con los enlaces simbólicos ya que solo el propietario verdadero tiene un apuntador al nodo-i:
  - · Los usuarios enlazados al archivo solo tienen nombres de rutas de acceso y no apuntadores a nodo-i.
  - · Cuando el propietario elimina un archivo, este se destruye.

Desventajas de la segunda solución:

- El principal problema es su costo excesivo, especialmente en accesos a disco, puesto que se debe leer el archivo que contiene la ruta de acceso, analizarla y seguirla componente a componente hasta alcanzar el nodo-i.
- Se precisa un nodo-i adicional por cada enlace simbólico y un bloque adicional en disco para almacenar la ruta de acceso.
- Los archivos pueden tener dos o más rutas de acceso, debido a lo cual, en búsquedas genéricas se podría encontrar el mismo archivo por distintas rutas y tratárselo como si fueran archivos distintos.

Los enlaces simbólicos tienen la ventaja de que se pueden utilizar para enlazar archivos en otras máquinas, en cualquier parte del mundo; se debe proporcionar solo la dirección de la red de la máquina donde reside el archivo y su ruta de acceso en esa máquina.

4.6.4 Administración del Espacio en Disco

Existen dos *estrategias generales* para almacenar un archivo de "n" bytes [23, Tanenbaum]:

Asignar "n" bytes consecutivos de espacio en el disco:

- Tiene el problema de que si un archivo crece será muy probable que deba desplazarse en el disco, lo que puede afectar seriamente al rendimiento.

Dividir el archivo en cierto número de bloques (no necesariamente) adyacentes:

- Generalmente los sistemas de archivos utilizan esta estrategia con bloques de tamaño fijo.

Tamaño del bloque:

Dada la forma en que están organizados los bloques, el sector, la pista y el cilindro son los candidatos obvios como unidades de asignación.

Si se tiene una *unidad de asignación grande*, como un cilindro, esto significa que cada archivo, inclusive uno pequeño, ocupará todo un cilindro; con esto se desperdicia espacio de almacenamiento en disco.

Si se utiliza una *unidad de asignación pequeña*, como un sector, implica que cada archivo constará de muchos bloques; con esto su lectura generará muchas operaciones de e / s afectando la performance.

Lo anterior indica que *la eficiencia en tiempo y espacio tienen un conflicto inherente.*

Generalmente se utilizan como solución de compromiso bloques de 1/2 k, 1k, 2k o 4k.[8]

Hay que recordar que el *tiempo de lectura de un bloque de disco* es la suma de los tiempos de:

[8] Ver Figura 4.8 de la página 137 [23, Tanenbaum].

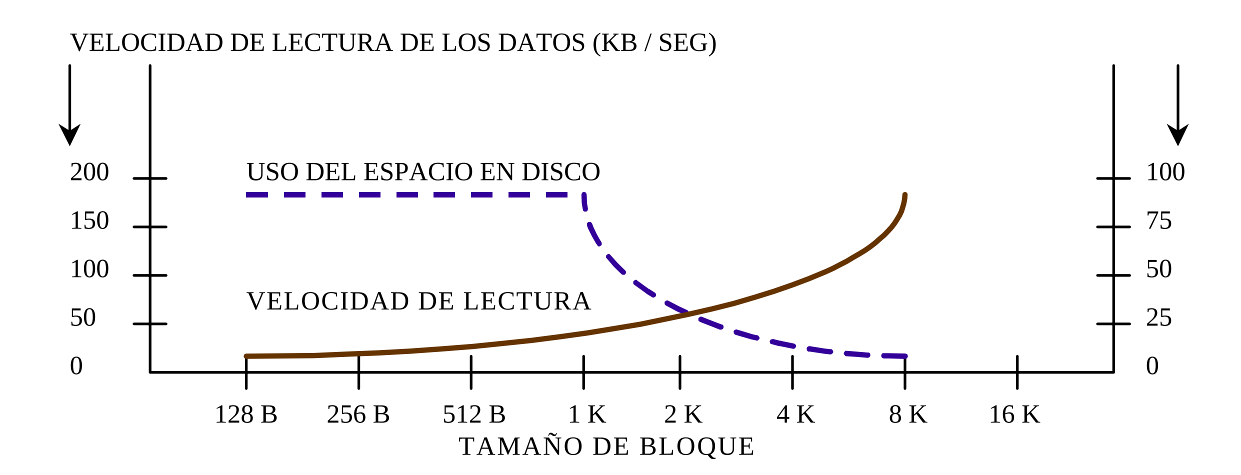


Figura 4.8: Representación de la velocidad de lectura y del uso del espacio en disco en función del tamaño de bloque.

- Búsqueda.
- Demora rotacional.
- Transferencia.

Registro de los bloques libres:
Se utilizan por lo general dos métodos:

- La lista de bloques libres como lista ligada.
- Un mapa de bits.

Lista ligada de bloques de disco:

- Cada bloque contiene tantos números de bloques libres como pueda.
- Los bloques libres se utilizan para contener a la lista de bloques libres.

Mapa de bits:

- Un disco con "n" bloques necesita un mapa de bits con "n" bits.
- Los bloques libres se representa con "1" y los asignados con "0" (o viceversa).
- Generalmente este método es preferible cuando existe espacio suficiente en la memoria principal para contener *completo el mapa de bits.*

Disk quotas:

Para evitar que los usuarios se apropien de un espacio excesivo en disco, los S. O. multiusuario proporcionan generalmente un mecanismo para establecer las cuotas en el disco.

La idea es que:

- Un administrador del sistema asigne a cada usuario una proporción máxima de archivos y bloques.
- El S. O. garantice que los usuarios no excedan sus cuotas.

Un mecanismo utilizado es el siguiente:

- Cuando un usuario **abre un archivo**:
  - Se localizan los atributos y direcciones en disco.
  - Se colocan en una tabla de archivos abiertos en la memoria principal.
  - Uno de los atributos indica el propietario del archivo; cualquier aumento del tamaño del archivo se carga a la cuota del propietario.
  - Una segunda tabla contiene el registro de las cuotas para cada uno de los usuarios que tengan un archivo abierto en ese momento, aún cuando el archivo lo haya abierto otro usuario.
- Cuando se escribe una **nueva entrada en la tabla de archivos abiertos**:
  - Se introduce un apuntador al registro de la cuota del propietario para localizar los límites.
- Cuando se **añade un bloque a un archivo**:
  - Se incrementa el total de bloques cargados al propietario.
  - Se verifica este valor contra los límites estricto y flexible (el primero no se puede superar, el segundo sí).
  - También se verifica el número de archivos.

4.6.5 Confiabilidad del Sistema de Archivos

Es necesario proteger la información alojada en el sistema de archivos, efectuando los resguardos correspondientes [23, Tanenbaum].

De esta manera se evitan las consecuencias generalmente catastróficas de la pérdida de los sistemas de archivos.

Las pérdidas se pueden deber a problemas de hardware, software, hechos externos, etc.

Manejo de un bloque defectuoso:

Se utilizan soluciones por hardware y por software.

La **solución en hardware**:

- Consiste en dedicar un sector del disco a la *lista de bloques defectuosos*.
- Al inicializar el controlador por primera vez:
  - Lee la "*lista de bloques defectuosos*".
  - Elige un bloque (o pista) de reserva para reemplazar los defectuosos.

- Registra la asociación en la lista de bloques defectuosos.
- En lo sucesivo, las solicitudes del bloque defectuoso utilizarán el de repuesto.

La **solución en software**:

- Requiere que el usuario o el sistema de archivos construyan un *archivo con todos los bloques defectuosos.*
- Se los elimina de la "*lista de bloques libres*".
- Se crea un "*archivo de bloques defectuosos*":
  - Esta constituido por los bloques defectuosos.
  - No debe ser leído ni escrito.
  - No se debe intentar obtener copias de respaldo de este archivo.

Respaldos (copias de seguridad o de back-up):
Es muy importante respaldar los archivos con frecuencia.

Los respaldos pueden consistir en efectuar copias completas del contenido de los discos (flexibles o rígidos).

Una estrategia de respaldo consiste en dividir los discos en *áreas de datos y áreas de respaldo*, utilizándolas de a pares:

- Se desperdicia la mitad del almacenamiento de datos en disco para respaldo.
- Cada noche (o en el momento que se establezca), la parte de datos de la unidad 0 se copia a la parte de respaldo de la unidad 1 y viceversa.

Otra estrategia es el vaciado por incrementos o *respaldo incremental*:

- Se obtiene una copia de respaldo periódicamente (por ej.: una vez por mes o por semana), llamada copia total.
- Se obtiene una copia diaria solo de aquellos archivos modificados desde la última copia total; en estrategias mejoradas, se copian solo aquellos archivos modificados desde la última vez que dichos archivos fueron copiados.
- Se debe mantener en el disco información de control como una "*lista de los tiempos de copiado*" de cada archivo, la que debe ser actualizada cada vez que se obtienen copias de los archivos y cada vez que los archivos son modificados.
- Puede requerir una gran cantidad de cintas de respaldo dedicadas a los respaldos diarios entre respaldos completos.

Consistencia del sistema de archivos:
Muchos sistemas de archivos leen bloques, los modifican y escriben en ellos después.

Si el sistema falla antes de escribir en los bloques modificados, el sistema de archivos puede quedar en un "*estado inconsistente*".

La inconsistencia es particularmente crítica si alguno de los bloques afectados son:

- Bloques de nodos-i.
- Bloques de directorios.
- Bloques de la lista de bloques libres.

La mayoría de los sistemas dispone de un programa utilitario que verifica la *consistencia del sistema de archivos*:

- Se pueden ejecutar al arrancar el sistema o a pedido.
- Pueden actuar sobre todos o algunos de los discos.
- Pueden efectuar verificaciones a nivel de bloques y a nivel de archivos.
- La consistencia del sistema de archivos no asegura la consistencia interna de cada archivo, respecto de su contenido.
- Generalmente pueden verificar también el sistema de directorios y / o de bibliotecas.

Generalmente los utilitarios utilizan dos tablas:

- Tabla de bloques en uso.
- Tabla de bloques libres.
- Cada bloque debe estar referenciado en una de ellas.

Si un bloque no aparece en ninguna de las tablas se trata de una falla llamada *bloque faltante*:

- No produce daños pero desperdicia espacio en disco.
- Se soluciona añadiendo el bloque a la tabla de bloques libres.

También podría detectarse la situación de falla debida a un *bloque referenciado dos veces en la tabla de bloques libres*:

- Esta falla no se produce en los sistemas de archivos basados en mapas de bits, sí en los basados en tablas o listas.
- La solución consiste en depurar la tabla de bloques libres.

Una *falla muy grave* es que *el mismo bloque de datos aparezca referenciado dos o más veces en la tabla de bloques en uso*:

- Como parte del mismo o de distintos archivos.
- Si uno de los archivos se borra, el bloque aparecería en la tabla de bloques libres y también en la de bloques en uso.
- Una solución es que el verificador del sistema de archivos:

  - Asigne un bloque libre.
  - Copie en el bloque libre el contenido del bloque conflictivo.
  - Actualice las tablas afectando el bloque copia a alguno de los archivos.
  - Agregue el bloque conflictivo a la tabla de bloques libres.
  - Informe al usuario para que verifique el daño detectado y la solución dada.

Otro error posible es que *un bloque esté en la tabla de bloques en uso y en la tabla de bloques libres*:

- Se soluciona eliminándolo de la tabla de bloques libres.

Las verificaciones de directorios incluyen controles como:

- Número de directorios que apuntan a un nodo-i con los contadores de enlaces almacenados en los propios nodos-i; en un sistema consistente de archivos deben coincidir.

Una posible falla es que *el contador de enlaces sea mayor que el número de entradas del directorio*:

- Aunque se eliminaran todos los archivos de los directorios el contador sería distinto de cero y no se podría eliminar el nodo-i.
- No se trata de un error serio pero produce desperdicio de espacio en disco con archivos que no se encuentran en ningún directorio.
- Se soluciona haciendo que el contador de enlaces en el nodo-i tome el valor correcto; si el valor correcto es 0, el archivo debe eliminarse.

Otro tipo de *error* es potencialmente *catastrófico*:

- Si dos entradas de un directorio se enlazan a un archivo, pero el nodo-i indica que solo existe un enlace, entonces, al eliminar cualquiera de estas entradas de directorio, el contador del nodo-i tomará el valor 0.
- Debido al valor 0 el sistema de archivos lo señala como no utilizado y libera todos sus bloques.
- Uno de los directorios apunta hacia un nodo-i no utilizado, cuyos bloques se podrían asignar entonces a otros archivos.
- La solución es forzar que el contador de enlaces del nodo-i sea igual al número de entradas del directorio.

También se pueden hacer *verificaciones heurísticas*, por ej.:

- Cada nodo-i tiene un modo, pero algunos modos son válidos aunque extraños:
  - Ej.: Se prohibe el acceso al propietario y todo su grupo, pero se permite a los extraños leer, escribir y ejecutar el archivo.
  - La verificación debería detectar e informar de estas situaciones.
- Se debería informar como sospechosos aquellos directorios con excesivas entradas, por ej., más de mil.

4.6.6 Desempeño del Sistema de Archivos

El acceso al disco es mucho más lento que el acceso a la memoria:

- Los tiempos se miden en milisegundos y en nanosegundos respectivamente.
- Se debe reducir el número de accesos a disco.

La técnica más común para *reducir los accesos a disco* es el **bloque caché** o **buffer caché [23, Tanenbaum]**:

- Se utiliza el término *ocultamiento* para esta técnica (del francés "cacher": ocultar).
- Un caché es una colección de bloques que pertenecen desde el punto de vista lógico al disco, pero que se mantienen en memoria por razones de rendimiento.

Uno de los algoritmos más comunes para la administración del caché es el siguiente:

- Verificar todas las solicitudes de lectura para saber si el bloque solicitado se encuentra en el caché.
- En caso afirmativo, se satisface la solicitud sin un acceso a disco.
- En caso negativo, se lee para que ingrese al caché y luego se copia al lugar donde se necesite.
- Cuando hay que cargar un bloque en un caché totalmente ocupado:
  - Hay que eliminar algún bloque y volverlo a escribir en el disco en caso de que haya sido modificado luego de haberlo traído del disco.
  - Se plantea una situación muy parecida a la paginación y se resuelve con algoritmos similares.

Se debe considerar la *posibilidad de una falla total del sistema y su impacto en la consistencia del sistema de archivos*:

- Si un bloque crítico, como un bloque de un nodo-i, se lee en el caché y se modifica, sin volverse a escribir en el disco, una falla total del sistema dejará al sistema de archivos en un estado inconsistente.

Se deben tener en cuenta los siguientes factores:

- ¿ Es posible que el bloque modificado se vuelva a necesitar muy pronto ?:
  - Los bloques que se vayan a utilizar muy pronto, como un bloque parcialmente ocupado que se está escribiendo, deberían permanecer un "*largo tiempo*".
- ¿ Es esencial el bloque para la consistencia del sistema de archivos ?:

  - Si es esencial (generalmente lo será si no es bloque de datos) y ha sido modificado, debe escribirse en el disco de inmediato:
    * Se reduce la probabilidad de que una falla total del sistema haga naufragar al sistema de archivos.
    * Se debe elegir con cuidado el orden de escritura de los bloques críticos.

- No es recomendable mantener los bloques de datos en el caché durante mucho tiempo antes de reescribirlos.

La solución de algunos S. O. consiste en tener una *llamada al sistema que fuerza una actualización general* a intervalos regulares de algunos segundos (por ej. 30).

Otra solución consiste en escribir los bloques modificados (del caché) al disco, tan pronto como haya sido escrito (el caché):

- Se dice que se trata de **cachés de escritura**.
- Requiere más e / s que otros tipos de cachés.

Una técnica importante para *aumentar el rendimiento de un sistema de archivos* es la reducción de la cantidad de movimientos del brazo del disco (mecanismo de acceso):

- Se deben colocar los bloques que probablemente tengan un acceso secuencial, próximos entre sí, preferentemente en el mismo cilindro.
- Los nodos-i deben estar a mitad del disco y no al principio, reduciendo a la mitad el tiempo promedio de búsqueda entre el nodo-i y el primer bloque del archivo.

4.7 Descriptor de Archivos

El *descriptor de archivos o bloque de control de archivos* es un bloque de control que contiene información que el sistema necesita para administrar un archivo [7, Deitel].

Es una estructura muy dependiente del sistema.

Puede incluir la siguiente información:

- Nombre simbólico del archivo.
- Localización del archivo en el almacenamiento secundario.
- Organización del archivo (método de organización y acceso).
- Tipo de dispositivo.
- Datos de control de acceso.
- Tipo (archivo de datos, programa objeto, programa fuente, etc.).
- Disposición (permanente contra temporal).
- Fecha y tiempo de creación.

- Fecha de destrucción.
- Fecha de la última modificación.
- Suma de las actividades de acceso (número de lecturas, por ejemplo).

Los descriptores de archivos suelen mantenerse en el almacenamiento secundario; se pasan al almacenamiento primario al abrir el archivo.

El descriptor de archivos es controlado por el *sistema de archivos*; el usuario puede no hacer referencia directa a él.

4.8 Seguridad

Los sistemas de archivos generalmente contienen información muy valiosa para sus usuarios, razón por la que los sistemas de archivos deben protegerla [23, Tanenbaum].

4.8.1 El Ambiente de Seguridad.

Se entenderá por seguridad a los problemas generales relativos a la garantía de que los archivos no sean leídos o modificados por personal no autorizado; esto incluye aspectos técnicos, de administración, legales y políticos.

Se consideraran mecanismos de protección a los mecanismos específicos del sistema operativo utilizados para resguardar la información de la computadora.

La frontera entre *seguridad* y *mecanismos de protección* no está bien definida.

Dos de las más importantes facetas de la seguridad son:

- La pérdida de datos.
- Los intrusos.

Algunas de las *causas más comunes de la pérdida de datos* son:

- Actos y hechos diversos, como incendios, inundaciones, terremotos, guerras, revoluciones, roedores, etc.
- Errores de hardware o de software, como fallas en la cpu, discos o cintas ilegibles, errores de telecomunicación, errores en los programas, etc.
- Errores humanos, por ej., entrada incorrecta de datos, mal montaje de cintas o discos, ejecución incorrecta de programas, pérdida de cintas o discos, etc.

La mayoría de estas causas se pueden enfrentar con el mantenimiento de los *respaldos (back-ups)* adecuados; debería haber copias en un lugar alejado de los datos originales.

Respecto del problema de los **intrusos**, se los puede clasificar como:

- **Pasivos**: solo desean leer archivos que no están autorizados a leer.
- **Activos**: desean hacer cambios no autorizados a los datos.

Para diseñar un sistema seguro contra intrusos:

- Hay que tener en cuenta el tipo de intrusos contra los que se desea tener protección.
- Hay que ser consciente de que la cantidad de esfuerzo que se pone en la seguridad y la protección depende claramente de quién se piensa sea el enemigo.

Algunos *tipos de intrusos* son los siguientes:

- Curiosidad casual de usuarios no técnicos.
- Conocidos (técnicamente capacitados) husmeando.
- Intentos deliberados por hacer dinero.
- Espionaje comercial o militar.

Otro aspecto del problema de la seguridad es la **privacía**:

- Protección de las personas respecto del mal uso de la información en contra de uno mismo.
- Implica aspectos legales y morales.

También debe señalarse la posibilidad del ataque del **caballo de Troya**:

- Modificar un programa normal para que haga cosas adversas además de su función usual.
- Arreglar las cosas para que la víctima utilice la versión modificada.

Además debe considerarse la posibilidad de ataques al estilo del **gusano de Internet**:

- Fue liberado por Robert Tappan Morris el 02/11/88 e hizo que se bloquearan la mayoría de los sistemas Sun y Vax de Internet (fue descubierto y condenado).
- Constaba de un programa arrancador y del gusano propiamente dicho.
- Utilizaba fallas se seguridad del Unix y de los programas Finger y Sendmail de Internet.

Una forma de probar la seguridad de un sistema es contratar un grupo de expertos en seguridad, conocido como el *equipo tigre* o *equipo de penetración*, cuyo objetivo es intentar penetrar el sistema de seguridad para descubrir sus falencias y proponer soluciones.

Otro aspecto importante de la seguridad consiste en *no subestimar los problemas* que puede causar el personal.

4.8.2 Virus

Los virus computacionales:

- Constituyen una categoría especial de ataque.
- Son un enorme problema para muchos usuarios.
- Son fragmentos de programas que se añaden a programas legítimos con la intención de infectar a otros.
- Un virus difiere de un gusano en lo siguiente:
  - Un virus está a cuestas de un programa existente.
  - Un gusano es un programa completo en sí mismo.
- Los virus y los gusanos intentan diseminarse y pueden crear un daño severo.
- Generalmente se propagan a través de copias ilegítimas de programas.
- Comúnmente los virus se ejecutan e intentan reproducirse cada vez que se ejecuta el programa que los aloja.
- Frecuentemente los problemas con los virus son más fáciles de evitar que de curar:
  - Utilizar software original adquirido en comercios respetables.
  - No utilizar copias "piratas".
  - Efectuar controles rigurosos y frecuentes con programas antivirus actualizados.
  - Trabajar con metodología y disciplina rigurosa en el intercambio de discos y en las copias a través de redes de comunicación de datos.

4.8.3 Principios del Diseño Para la Seguridad

El diseño del sistema debe ser público, ya que pensar que el intruso no conocerá la forma de funcionamiento del sistema es un engaño.

El *estado predefinido* debe ser el *de no acceso*, dado que los errores en donde se niega el acceso válido se reportan más rápido que los errores en donde se permite el acceso no autorizado.

Verificar la *autorización actual*:

- El sistema no debe:
  - Verificar el permiso.
  - Determinar que el acceso está permitido.
  - Abandonar esta información para su uso posterior.
- El sistema tampoco debe:

- Verificar el permiso al abrir un archivo y no después de abrirlo, pues un acceso habilitado permanecería como válido aunque haya cambiado la protección del archivo.

Dar a cada proceso el *mínimo privilegio* posible, lo que implica un esquema de "*protección de grano fino*".

El *mecanismo de protección* debe ser simple, uniforme e integrado hasta las capas más bajas del sistema:

- Dotar de seguridad a un sistema inseguro es casi imposible.
- La seguridad no es una característica que se pueda añadir fácilmente.

El esquema de seguridad debe ser *sicológicamente aceptable*:

- Los usuarios no deben sentir que la protección de sus archivos les implica demasiado trabajo:
  - Podrían dejar de proteger sus archivos.
  - Se quejarían en caso de problemas.
  - No aceptarían fácilmente su propia culpa.

4.8.4 Autentificación del Usuario

Muchos esquemas de protección se basan en la hipótesis de que *el sistema conoce la identidad de cada usuario.*

La identificación de los usuarios se conoce como la *autentificación de los usuarios.*

Muchos métodos de autentificación se basan en:

- La identificación de algo *conocido por el usuario.*
- Algo que *posee el usuario.*
- Algo que *es el usuario.*

4.8.5 Contraseñas

Son la forma de autentificación más utilizada.

Son de fácil comprensión e implementación.

Deben *almacenarse cifradas (encriptadas).*

Se deben prever intentos de penetración consistentes en *pruebas de combinaciones* de nombres y contraseñas.

Si las contraseñas fueran de 7 caracteres elegidos al azar de los 95 caracteres ASCII que se pueden imprimir:

- El espacio de búsqueda sería de 95^{7}, alrededor de 7×10^{13}.
- A 1.000 ciframientos por segundo tomaría 2.000 años construir la lista a verificar contra el archivo de contraseñas.

Una *mejora al esquema de contraseñas* consiste en:

- Asociar un número aleatorio de "n" bits a cada contraseña.
- El número aleatorio se modifica al cambiar la contraseña.
- El número se guarda en el archivo de contraseñas en forma no cifrada.
- Se concatenan la contraseña y el número aleatorio y se cifran juntos.
- El resultado cifrado se almacena en el archivo de contraseñas.
- Se aumenta por 2^n el espectro de búsqueda: a esto se llama *salar el archivo de contraseñas.*

Una protección adicional consiste en hacer ilegible el archivo de contraseñas encriptadas.

Otra protección adicional consiste en que el sistema sugiera a los usuarios contraseñas generadas según *ciertos criterios*; con esto se evita que el usuario elija contraseñas muy sencillas.

También es conveniente que el sistema obligue al usuario a *cambiar sus contraseñas con regularidad*; se puede llegar a la *contraseña de una sola vez.*

Una *variante* de la idea de contraseña es *solicitar al usuario respuestas* sobre información de contexto que debe conocer.

Otra variante es la de *reto-respuesta*:

- Se acuerdan con el usuario **algoritmos** (por ejemplo formulas matemáticas) que se utilizarán según el día y / o la hora.
- Cuando el usuario se conecta:
  - El sistema suministra un argumento.
  - El usuario debe responder con el resultado correspondiente al algoritmo vigente ese día a esa hora.

4.8.6 Identificación Física

Una posibilidad es la verificación de si el usuario tiene *cierto elemento* (generalmente una tarjeta plástica con una banda magnética), que generalmente se combina con una contraseña.

Otro aspecto consiste en la medición de *características físicas* difíciles de reproducir:

- Huellas digitales o vocales.
- Firmas.
- Longitud de los dedos de las manos.

4.8.7 Medidas Preventivas

Limitar los intentos de acceso fallidos y registrarlos.

Registrar todos los accesos.

Tender trampas para atrapar a los intrusos.

4.9 Mecanismos de Protección

4.9.1 Dominios de Protección

Muchos *objetos del sistema* necesitan protección, tales como la cpu, segmentos de memoria, unidades de disco, terminales, impresoras, procesos, archivos, bases de datos, etc. [23, Tanenbaum].

Cada objeto se referencia por un *nombre* y tiene habilitadas un conjunto de *operaciones* sobre él.

Un **dominio** es un *conjunto de parejas (objeto, derechos):*

- Cada pareja determina:
  - Un objeto.
  - Un subconjunto de las operaciones que se pueden llevar a cabo en él.

Un **derecho** es el *permiso para realizar alguna de las operaciones.*

Es posible que un objeto se encuentre en varios dominios con "*distintos*" derechos en cada dominio.

Un **proceso** se *ejecuta en alguno de los dominios de protección:*

- Existe una colección de objetos a los que puede tener acceso.
- Cada objeto tiene cierto conjunto de derechos.

Los procesos pueden alternar entre los dominios durante la ejecución.

Una llamada al S. O. provoca una *alternancia de dominio.*

En algunos S. O. los dominios se llaman **anillos**.

Una forma en la que el S. O. lleva un registro de los objetos que pertenecen a cada dominio es mediante una *matriz*:

- Los renglones son los dominios.
- Las columnas son los objetos.
- Cada elemento de la matriz contiene los derechos correspondientes al objeto en ese dominio, por ej.: leer, escribir, ejecutar.

4.9.2 Listas Para Control de Acceso

Las "*matrices de protección*" son muy grandes y con muchos lugares vacíos [23, Tanenbaum]:

- Desperdician espacio de almacenamiento.
- Existen métodos prácticos que almacenan solo los elementos no vacíos por filas o por columnas.

La *lista de control de acceso* (**ACL**: *access control list*):

- Asocia a *cada objeto una lista ordenada* con:
  - Todos los dominios que pueden tener acceso al objeto.
  - La forma de dicho acceso (ej: lectura (r), grabación (w), ejecución (x)).

Una *forma de implementar* las ACL consiste en:

- Asignar tres bits (r, w, x) para cada archivo, para:
  - El propietario, el grupo del propietario y los demás usuarios.
- Permitir que el propietario de cada objeto pueda modificar su ACL en cualquier momento:
  - Permite prohibir accesos antes permitidos.

4.9.3 Posibilidades

La *matriz de protección* también puede dividirse por renglones [23, Tanenbaum]:

- Se le asocia a *cada proceso una lista de objetos* a los cuales puede tener acceso.
- Se le indican las *operaciones permitidas* en cada uno.
- Esto *define su dominio.*

La lista de objetos se denomina lista de posibilidades y los elementos individuales se llaman posibilidades.

Cada **posibilidad** tiene:

- Un *campo tipo*:
  - Indica el tipo del objeto.
- Un *campo derechos*:
  - Mapa de bits que indica las operaciones básicas permitidas en este tipo de objeto.

- Un *campo objeto*:
    - Apuntador al propio objeto (por ej.: su número de nodo-i).

Las *listas de posibilidades* son a su vez **objetos** y se les puede apuntar desde otras listas de posibilidades; esto facilita la existencia de **subdominios compartidos**.

Las listas de posibilidades o *listas-c* deben ser protegidas del manejo indebido por parte del usuario.

Los principales **métodos de protección** son:

- *Arquitectura marcada:*
    - Necesita un diseño de hardware en el que cada palabra de memoria tiene un bit adicional:
        * Indica si la palabra contiene una posibilidad o no.
        * Solo puede ser modificado por el S. O.
- *Lista de posibilidades dentro del S. O.:*
    - Los procesos hacen referencia a las posibilidades mediante su número.
- *Lista de posibilidades cifrada dentro del espacio del usuario:*
    - Cada posibilidad está cifrada con una clave secreta desconocida por el usuario.
    - Muy adecuado para sistemas distribuidos.

Generalmente las **posibilidades** tienen *derechos genéricos aplicables a todos los objetos*, por ej.:

- Copiar posibilidad:
    - Crear una nueva posibilidad para el mismo objeto.
- Copiar objeto:
    - Crear un duplicado del objeto con una nueva posibilidad.
- Eliminar posibilidad:
    - Eliminar un dato dentro de la lista-c sin afectar al objeto.
- Destruir objeto:
    - Eliminar en forma permanente un objeto y una posibilidad.

Muchos sistemas con posibilidades se organizan como una colección de módulos con *módulos administradores de tipos* para cada tipo de objeto y entonces es esencial que el módulo administrador de tipos pueda hacer más cosas con la posibilidad que un proceso ordinario.

Se utiliza la técnica de *amplificación de derechos*:

- Los administradores de tipo obtienen una plantilla de derechos que les da *más derechos sobre un objeto* de los que permitía la propia lista de posibilidades.

4.9.4 Modelos de Protección

Las matrices de protección no son estáticas sino dinámicas [23, Tanenbaum].

Se pueden identificar seis *operaciones primitivas* en la *matriz de protección*:

- Crear objeto.
- Eliminar objeto.
- Crear dominio.
- Eliminar dominio.
- Insertar derecho.
- Eliminar derecho.

Las primitivas se pueden combinar en *comandos de protección*, que pueden ser ejecutados por los programas del usuario para *modificar la matriz de protección.*

En cada momento, la matriz de protección determina *lo que puede hacer un proceso* en cualquier momento; no determina lo que no está autorizado a realizar.

La matriz es impuesta por el sistema.

La *autorización* tiene que ver con la *política de administración.*

4.9.5 Control de Acceso Por Clases de Usuarios

Una matriz de control de acceso puede llegar a ser tan grande que resulte impráctico mantenerla [7, Deitel].

Una técnica que *requiere menos espacio* es *controlar el acceso a varias clases de usuarios.*

Un ejemplo de *esquema de clasificación* es el siguiente:

- *Propietario:*
  - Suele ser el usuario que creó el archivo.
- *Usuario especificado:*
  - El propietario especifica quién más puede usar el archivo.
- *Grupo o proyecto:*

  - Los diferentes miembros de un grupo de trabajo sobre un proyecto, acceden a los diferentes archivos relacionados con el proyecto.

- *Público:*
  - Un archivo público puede ser accedido por cualquier usuario de la computadora.
  - Generalmente permite leer o ejecutar pero no escribir sobre el archivo.

4.10 Respaldo y Recuperación

La destrucción de la información, ya sea accidental o intencional, es una realidad y tiene distintas causas [7, Deitel]:

- Fallas de hardware y de software.
- Fenómenos meteorológicos atmosféricos.
- Fallas en el suministro de energía.
- Incendios e inundaciones.
- Robos, vandalismo (incluso terrorismo).
- Etc.

Esta posible destrucción de la información debe ser tenida en cuenta por:

- Los sistemas operativos en general.
- Los sistemas de archivos en particular.

Una técnica muy usada para asegurar la disponibilidad de los datos es *realizar respaldos periódicos*:

- Hacer con regularidad una o más copias de los archivos y colocarlas en lugar seguro.
- Todas las actualizaciones realizadas luego del último respaldo pueden perderse.

Otra técnica es *pasar todas las transacciones a un archivo*, copiándolas en otro disco:

- Genera una redundancia que puede ser costosa.
- En caso de fallas en el disco principal, puede reconstruirse todo el trabajo perdido si el disco de reserva no se dañó también.

También existe la posibilidad del *respaldo incremental*:

- Durante una sesión de trabajo los archivos modificados quedan marcados.
- Cuando un usuario se retira del sistema (deja de trabajar), un proceso del sistema efectúa el respaldo de los archivos marcados.

Se debe tener presente que es muy difícil garantizar una seguridad absoluta de los archivos.

Capítulo 5

Entrada / Salida

5.1 Introducción

Una de las funciones principales de un S. O. es el *control de todos los dispositivos de e / s* de la computadora [23, Tanenbaum].

Las principales **funciones** relacionadas son:

- Enviar comandos a los dispositivos.
- Detectar las interrupciones.
- Controlar los errores.
- Proporcionar una interfaz entre los dispositivos y el resto del sistema:
  - Debe ser sencilla y fácil de usar.
  - Debe ser la misma (preferentemente) para todos los dispositivos (*independencia del dispositivo*).

El código de e / s representa una fracción significativa del S. O.

El uso inapropiado de los dispositivos de e / s frecuentemente genera ineficiencias del sistema, lo que afecta la performance global.

5.2 Principios del Hardware de E / S

El enfoque que se considerará tiene que ver con la interfaz que desde el hardware se presenta al software: [23, Tanenbaum]

- Comandos que acepta el hardware.
- Funciones que realiza.
- Errores que puede informar.

5.2.1 Dispositivos de E / S

Se pueden **clasificar** en dos grandes categorías:

- *Dispositivos de bloque.*
- *Dispositivos de caracter.*

Las principales características de los *dispositivos de bloque* son:

- La información se almacena en bloques de tamaño fijo.
- Cada bloque tiene su propia dirección.
- Los tamaños más comunes de los bloques van desde los 128 bytes hasta los 1.024 bytes.
- Se puede leer o escribir en un bloque de forma independiente de los demás, en cualquier momento.
- Un ejemplo típico de dispositivos de bloque son los discos.

Las principales características de los *dispositivos de caracter* son:

- La información se transfiere como un flujo de caracteres, sin sujetarse a una estructura de bloques.
- No se pueden utilizar direcciones.
- No tienen una operación de búsqueda.
- Un ejemplos típico de dispositivos de caracter son las impresoras de línea, terminales, interfaces de una red, ratones, etc.

Algunos dispositivos no se ajustan a este esquema de clasificación, por ejemplo los relojes, que no tienen direcciones por medio de bloques y no generan o aceptan flujos de caracteres.

El *sistema de archivos* solo trabaja con *dispositivos de bloque abstractos*, por lo que encarga la parte *dependiente del dispositivo* a un software de menor nivel, el *software manejador del dispositivo*.

5.2.2 Controladores de Dispositivos

Las **unidades de e / s** generalmente constan de:

- Un *componente mecánico*.
- Un componente electrónico, el *controlador del dispositivo* o *adaptador*.

Muchos controladores pueden manejar más de un dispositivo.

El S. O. generalmente trabaja con el controlador y no con el dispositivo.

Los modelos más frecuentes de *comunicación entre la cpu y los controladores* son:

- Para la mayoría de las micro y mini computadoras:
  - Modelo de *bus del sistema*.
- Para la mayoría de los mainframes:
  - Modelo de varios buses y computadoras especializadas en e / s llamadas *canales de e / s*.

La *interfaz entre el controlador y el dispositivo* es con frecuencia de muy bajo nivel:

- La comunicación es mediante un *flujo de bits en serie* que:
  - Comienza con un **preámbulo**.
  - Sigue con una serie de bits (de un sector de disco, por ej.).
  - Concluye con una suma para verificación o un código corrector de errores.
- El preámbulo:
  - Se escribe al dar formato al disco.
  - Contiene el número de cilindro y sector, el tamaño de sector y otros datos similares.

El **controlador** debe:

- Convertir el flujo de bits en serie en un bloque de bytes.
- Efectuar cualquier corrección de errores necesaria.
- Copiar el bloque en la memoria principal.

Cada controlador posee **registros** que utiliza para comunicarse con la cpu:

- Pueden ser parte del espacio normal de direcciones de la memoria: *e / s mapeada a memoria*.
- Pueden utilizar un espacio de direcciones especial para la e / s, asignando a cada controlador una parte de él.

El S. O. realiza la e / s al escribir comandos en los registros de los controladores; los parámetros de los comandos también se cargan en los registros de los controladores.

Al *aceptar el comando*, la cpu puede dejar al controlador y dedicarse a otro trabajo.

Al *terminar el comando*, el controlador *provoca una interrupción* para permitir que el S. O.:

- Obtenga el control de la cpu.
- Verifique los resultados de la operación.

La cpu *obtiene los resultados* y el estado del dispositivo al leer uno o más bytes de información de los registros del controlador.

Ejemplos de *controladores*, sus *direcciones de e / s* y sus *vectores de interrupción* en la PC IBM pueden verse en la Tabla 5.1 de la página 158 [23, Tanenbaum].

Controlador de e / s	Dirección de e / s	Vector de interrupciones
Reloj	040 - 043	8
Teclado	060 - 063	9
Disco duro	320 - 32f	13
Impresora	378 - 37f	15
Disco flexible	3f0 - 3f7	14
Rs232 primario	3f8 - 3ff	12
Rs232 secundario	2f8 - 2ff	11

Tabla 5.1: Controladores de e / s, direcciones de e / s y vector de interrupciones.

5.2.3 Acceso Directo a Memoria (DMA)

Muchos controladores, especialmente los correspondientes a dispositivos de bloque, permiten el DMA.

Si se lee el disco *sin DMA*:

- El controlador lee en serie el bloque (uno o más sectores) de la unidad:
  - La lectura es bit por bit.
  - Los bits del bloque se graban en el buffer interno del controlador.
- Se calcula la suma de verificación para corroborar que no existen errores de lectura.
- El controlador provoca una interrupción.
- El S. O. lee el bloque del disco por medio del buffer del controlador:
  - La lectura es por byte o palabra a la vez.
  - En cada iteración de este ciclo se lee un byte o una palabra del registro del controlador y se almacena en memoria.
- Se *desperdicia tiempo* de la cpu.

DMA se ideó para liberar a la cpu de este trabajo de bajo nivel.

La cpu le proporciona al controlador:

- La dirección del bloque en el disco.
- La dirección en memoria adonde debe ir el bloque.
- El número de bytes por transferir.

Luego de que el controlador leyó todo el bloque del dispositivo a su buffer y de que corroboró la suma de verificación:

- Copia el primer byte o palabra a la memoria principal.

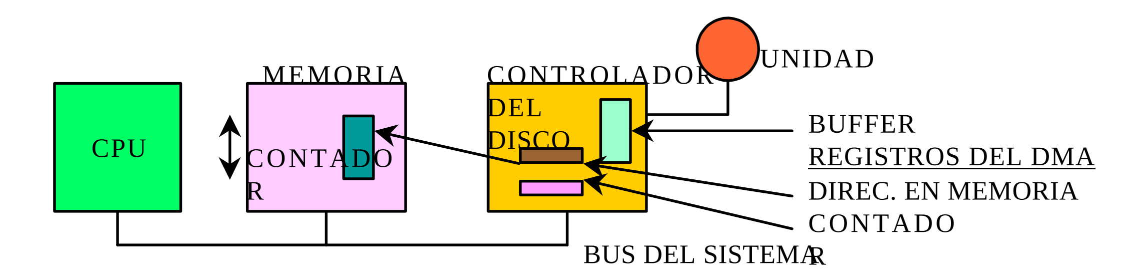


Figura 5.1: Un controlador realiza completamente una transferencia DMA.

- Lo hace en la dirección especificada por medio de la dirección de memoria de DMA.
- Incrementa la *dirección DMA* y decrementa el *contador DMA* en el número de bytes que acaba de transferir.
- Se repite este proceso hasta que el contador se anula y por lo tanto el controlador provoca una interrupción.
- Al iniciar su ejecución el S. O. luego de la interrupción provocada, no debe copiar el bloque en la memoria, porque ya se encuentra ahí.[1]

El controlador necesita un buffer interno porque una vez iniciada una transferencia del disco:

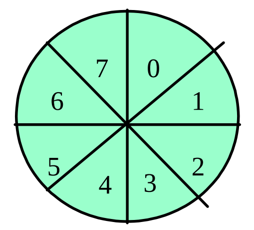
- Los bits siguen llegando del disco constantemente.
- No interesa si el controlador está listo o no para recibirlos.
- Si el controlador intentara escribir los datos en la memoria directamente:
  - Tendría que recurrir al bus del sistema para c / u de las palabras (o bytes) transferidas.
  - El bus podría estar ocupado por otro dispositivo y el controlador debería esperar.
  - Si la siguiente palabra llegara antes de que la anterior hubiera sido almacenada, el controlador la tendría que almacenar en alguna parte.

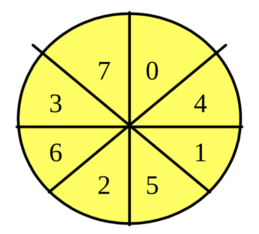
Si el bloque se guarda en un *buffer interno*:

- El bus no se necesita sino hasta que el DMA comienza.
- La transferencia DMA a la memoria ya no es un aspecto crítico del tiempo.

Los controladores simples no pueden atender la e / s simultánea:

[1] Ver Figura 5.1 de la página 159 [23, Tanenbaum].





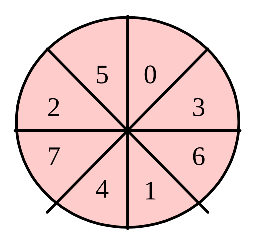


Figura 5.2: Factores de separación: sin separación, separación simple y separación doble.

- Mientras transfieren a la memoria, el sector que pasa debajo de la cabeza del disco se pierde; es decir que el bloque siguiente al recién leído se pierde.

- La lectura de una pista completa se hará en dos rotaciones completas, una para los bloques pares y otra para los impares.

- Si el tiempo necesario para una transferencia de un bloque del controlador a la memoria por medio del bus es mayor que el tiempo necesario para leer un bloque del disco:

  - Sería necesario leer un bloque y luego saltar dos o más bloques.
  - El *salto de bloques*:
    - Se ejecuta para darle tiempo al controlador para la transferencia de los datos a la memoria.
    - Se llama separación.
    - Al formatear el disco, los bloques se numeran tomando en cuenta el *factor de separación*.[2]
    - Esto permite al S. O.:
      - Leer los bloques con numeración consecutiva.
      - Conservar la máxima velocidad posible del hardware.

5.3 Principios del Software de E / S

La idea básica es organizar el software como *una serie de capas* donde [23, Tanenbaum]:

- Las *capas inferiores* se encarguen de ocultar las peculiaridades del hardware a las capas superiores.

- Las *capas superiores* deben presentar una interfaz agradable, limpia y regular a los usuarios.

[2] Ver Figura 5.2 de la página 160 [23, Tanenbaum].

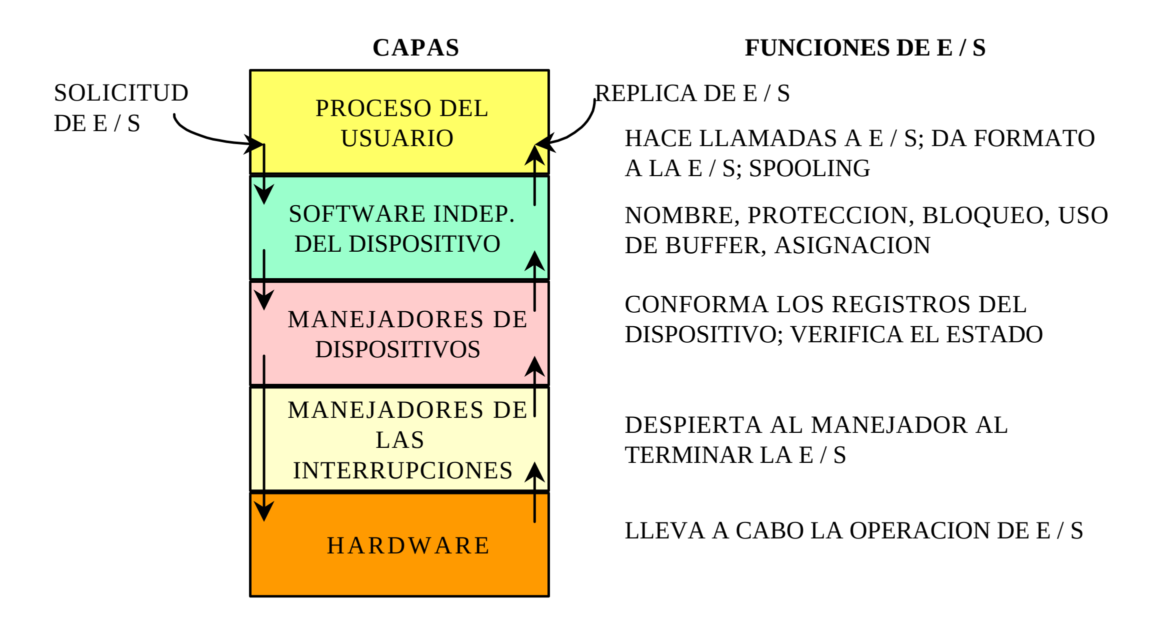


Figura 5.3: Capas del sistema de entrada / salida y las principales funciones de cada capa.

5.3.1 Objetivos del Software de E / S

Un concepto clave es la *independencia del dispositivo*:

- Debe ser posible escribir programas que se puedan utilizar con *archivos en distintos dispositivos*, sin tener que modificar los programas para cada tipo de dispositivo.

- El problema debe ser resuelto por el S. O.

El objetivo de lograr *nombres uniformes* está muy relacionado con el de independencia del dispositivo.

Todos los archivos y dispositivos adquieren direcciones de la misma forma, es decir mediante el *nombre de su ruta de acceso*.

Otro aspecto importante del software es el *manejo de errores de e / s*:

- Generalmente *los errores deben manejarse lo más cerca posible del hardware*.

- Solo si los niveles inferiores no pueden resolver el problema, se informa a los niveles superiores.

- Generalmente la recuperación se puede hacer en un nivel inferior y de forma transparente.

Otro aspecto clave son las *transferencias síncronas* (por bloques) o *asíncronas* (controlada por interruptores):

- La mayoría de la e / s es *asíncrona*: la cpu inicia la transferencia y realiza otras tareas hasta una interrupción.

- La programación es más fácil si la e / s es *síncrona* (por bloques): el programa se suspende automáticamente hasta que los datos estén disponibles en el buffer.

El S. O. se encarga de hacer que operaciones controladas por interruptores parezcan del tipo de bloques para el usuario.

También el S. O. debe administrar los **dispositivos compartidos** (ej.: discos) y los de **uso exclusivo** (ej.: impresoras).

Generalmente el software de e / s se estructura en capas:[3]

- Manejadores de interrupciones.
- Directivas de dispositivos.
- Software de S. O. independiente de los dispositivos.
- Software a nivel usuario.

5.3.2 Manejadores de Interrupciones

Las interrupciones deben ocultarse en el S. O.:

- Cada proceso que inicie una operación de e / s se bloquea hasta que termina la e / s y ocurra la interrupción.
- El procedimiento de interrupción realiza lo necesario para desbloquear el proceso que lo inicio.

5.3.3 Manejadores de Dispositivos

Todo el *código que depende de los dispositivos* aparece en los *manejadores de dispositivos.*

Cada controlador posee uno o más registros de dispositivos:

- Se utilizan para darle los comandos.
- Los manejadores de dispositivos proveen estos comandos y verifican su ejecución adecuada.

La labor de un *manejador de dispositivos* es la de:

- Aceptar las solicitudes abstractas que le hace el software independiente del dispositivo.
- Verificar la ejecución de dichas solicitudes.

Si al recibir una solicitud el manejador está ocupado con otra solicitud, agregara la nueva solicitud a una *cola de solicitudes pendientes.*

La solicitud de e / s, por ej. para un disco, se debe traducir de términos abstractos a *términos concretos:*

[3] Ver Figura 5.3 de la página 161 [23, Tanenbaum].

- El *manejador de disco* debe:
  - Estimar el lugar donde se encuentra en realidad el bloque solicitado.
  - Verificar si el motor de la unidad funciona.
  - Verificar si el brazo está colocado en el cilindro adecuado, etc.
  - Resumiendo: *debe decidir **cuáles** son las operaciones necesarias del controlador y su **orden**.*
  - Envía los comandos al controlador al escribir en los registros de dispositivo del mismo.
  - Frecuentemente el manejador del dispositivo se bloquea hasta que el controlador realiza cierto trabajo; una interrupción lo libera de este bloqueo.
  - Al finalizar la operación debe verificar los errores.
  - Si todo esta o.k. transferirá los datos al software independiente del dispositivo.
  - Regresa información de estado sobre los errores a quien lo llamó.
  - Inicia otra solicitud pendiente o queda en espera.

5.3.4 Software de E / S Independiente del Dispositivo

Funciones generalmente realizadas por el software independiente del dispositivo:

- Interfaz uniforme para los manejadores de dispositivos.
- Nombres de los dispositivos.
- Protección del dispositivo.
- Proporcionar un tamaño de bloque independiente del dispositivo.
- Uso de buffers.
- Asignación de espacio en los dispositivos por bloques.
- Asignación y liberación de los dispositivos de uso exclusivo.
- Informe de errores.

Las funciones básicas del software independiente del dispositivo son:

- Efectuar las funciones de e / s comunes a todos los dispositivos.
- Proporcionar una interfaz uniforme del software a nivel usuario.

El software independiente del dispositivo asocia los nombres simbólicos de los dispositivos con el nombre adecuado.

Un nombre de dispositivo determina de manera única el **nodo-i** de un archivo especial:

- Este nodo-i contiene el *número principal del dispositivo*, que se utiliza para localizar el manejador apropiado.

- El nodo-i contiene también el *número secundario de dispositivo*, que se transfiere como parámetro al manejador para determinar la unidad por leer o escribir.

El *software independiente del dispositivo* debe:

- Ocultar a los niveles superiores los diferentes tamaños de sector de los distintos discos.

- Proporcionar un tamaño uniforme de los bloques, por ej.: considerar varios sectores físicos como un solo bloque lógico.

5.3.5 Software de E / S en el Espacio del Usuario

La mayoría del software de e / s está dentro del S. O.

Una pequeña parte consta de bibliotecas ligadas entre sí con los programas del usuario. La biblioteca estándar de e / s contiene varios procedimientos relacionados con e / s y todos se ejecutan como parte de los programas del usuario.

Otra categoría importante de software de e / s a nivel usuario es el **sistema de spooling**.

El spooling es una forma de trabajar con los dispositivos de e /s de uso exclusivo en un *sistema de multiprogramación*:

- El ejemplo típico lo constituye la impresora de líneas.

- Los procesos de usuario no abren el archivo correspondiente a la impresora.

- Se crea un *proceso especial*, llamado *demonio* en algunos sistemas.

- Se crea un *directorio de spooling.*

Para *imprimir un archivo*:

- Un proceso genera todo el archivo por imprimir y lo coloca en el directorio de spooling.

- El proceso especial, único con permiso para utilizar el archivo especial de la impresora, debe imprimir los archivos en el directorio.

- Se evita el posible problema de tener un proceso de usuario que mantenga un recurso tomado largo tiempo.

Un esquema similar también es aplicable para la *transferencia de archivos* entre equipos conectados:

- Un usuario coloca un archivo en un directorio de spooling de la red.

- Posteriormente, el proceso especial lo toma y transmite. Un ej. son los sistemas de correo electrónico.

5.4 Discos - Hardware Para Discos

5.4.1 Discos

Las siguientes son las *principales ventajas* con respecto del uso de la memoria principal como almacenamiento [23, Tanenbaum]:

- Mucho mayor capacidad de espacio de almacenamiento.
- Menor precio por bit.
- La información no se pierde al apagar la computadora.

Un uso inapropiado de los discos puede generar ineficiencia, en especial en sistemas con multiprogramación.

5.4.2 Hardware Para Discos

Los discos están organizados en cilindros, pistas y sectores.

El número típico de sectores por pista varía entre 8 y 32 (o más).

Todos los sectores tienen igual número de bytes.

Los sectores cercanos a la orilla del disco serán mayores físicamente que los cercanos al anillo.

Un controlador puede realizar búsquedas en una o más unidades al mismo tiempo:

- Son las *búsquedas traslapadas.*
- Mientras el controlador y el software esperan el fin de una búsqueda en una unidad, el controlador puede iniciar una búsqueda en otra.

Muchos controladores pueden:

- Leer o escribir en una unidad.
- Buscar en otra.

Los controladores no pueden leer o escribir en dos unidades al mismo tiempo.

La capacidad de búsquedas traslapadas puede reducir considerablemente el tiempo promedio de acceso.

5.5 Operación de Almacenamiento de Disco de Cabeza Móvil

Los datos se graban en una serie de *discos magnéticos* o platos [7, Deitel].

El eje común de los discos gira a una velocidad del orden de las 4.000 o más revoluciones por minuto.

Se lee o escribe mediante una serie de *cabezas de lectura - escritura*:[4]

- Se dispone de una por cada superficie de disco.

[4] Ver Figura 5.4 de la página 166 [7, Deitel].

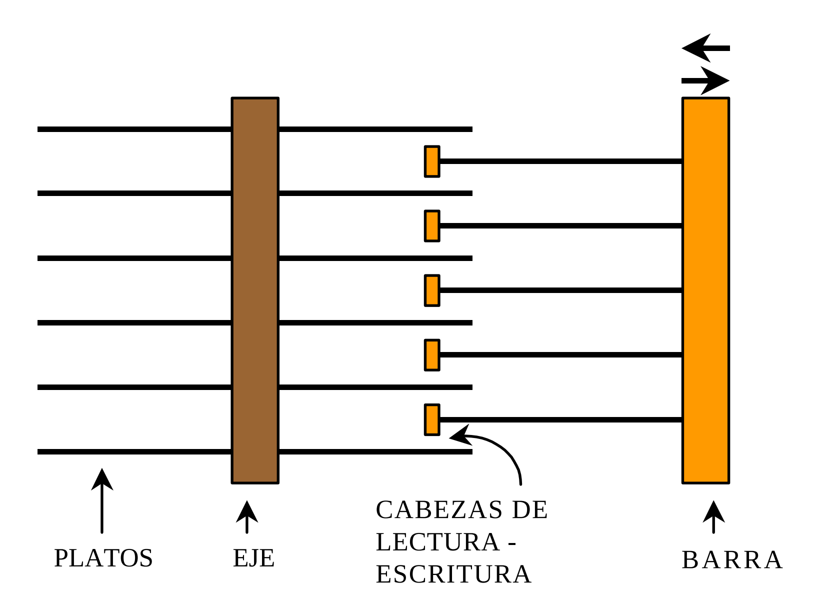


Figura 5.4: Esquema de un disco de cabeza móvil.

- Solo puede acceder a datos inmediatamente adyacentes a ella:
  - La parte de la superficie del disco de donde se leerá (o sobre la que se grabará) debe rotar hasta situarse inmediatamente debajo (o arriba) de la cabeza de lectura - escritura.
  - El tiempo de rotación desde la posición actual hasta la adyacente al cabezal se llama *tiempo de latencia.*

Todas las cabezas de lectura - escritura están montadas sobre una barra o conjunto de brazo móvil:

- Puede moverse hacia adentro o hacia afuera, en lo que se denomina *operación de búsqueda.*
- Para una posición dada, la serie de pistas accesibles forman un cilindro vertical.

A los *tiempos de búsqueda y de latencia* se debe agregar el *tiempo de transmisión propiamente dicha.*[5]

El *tiempo total de acceso* a un registro particular:

- Involucra movimientos mecánicos.
- Generalmente es del orden de centésimas de segundo, aunque el tiempo de latencia sea de algunas milésimas de segundo (7 a 12 aproximadamente).

[5] Ver Figura 5.5 de la página 167 [7, Deitel].

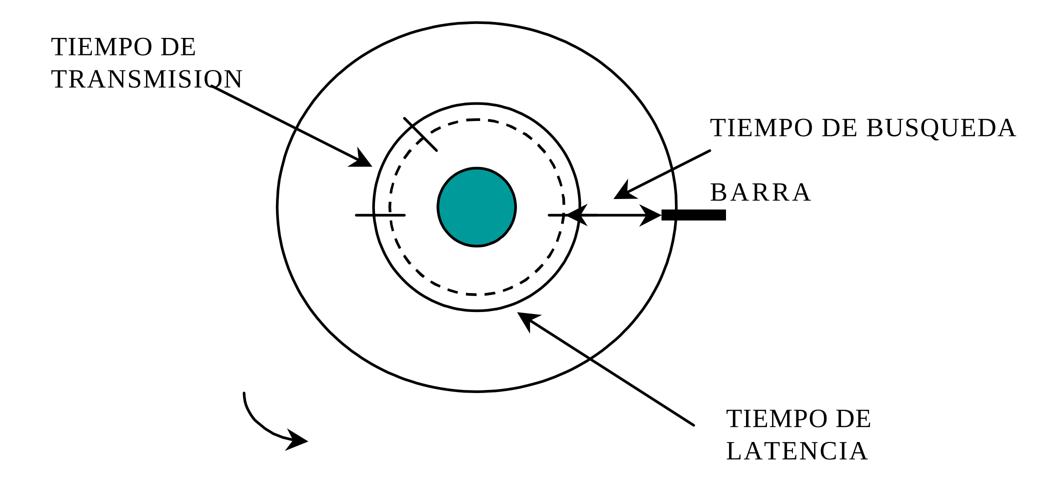


Figura 5.5: Componentes del acceso a un disco.

5.6 Algoritmos de Programación del Brazo del Disco

En la mayoría de los discos, el *tiempo de búsqueda* supera al de *retraso rotacional* y al de *transferencia [23, Tanenbaum]*, debido a ello, *la reducción del tiempo promedio de búsqueda puede mejorar en gran medida el rendimiento del sistema.*

Si el *manejador del disco* utiliza el algoritmo **primero en llegar primero en ser atendido (FCFS),** poco se puede hacer para mejorar el tiempo de búsqueda.

Es posible que mientras el brazo realiza una búsqueda para una solicitud, otros procesos generen otras solicitudes.

Muchos manejadores tienen una **tabla**:

- El índice es el número de cilindro.
- Incluye las *solicitudes pendientes para cada cilindro* enlazadas entre sí en una *lista ligada*.
- Cuando concluye una búsqueda, el *manejador del disco* tiene la opción de *elegir la siguiente solicitud* a dar paso:
  - Se atiende primero la solicitud más cercana, para *minimizar el tiempo de búsqueda*.
  - Este algoritmo se denomina **primero la búsqueda más corta** (**SSF: shortest seek first**).
  - Reduce a la mitad el número de movimientos del brazo en comparación con FCFS.

Ej. de *SSF*:

- Consideramos un disco de 40 cilindros.
- Se presenta una solicitud de lectura de un bloque en el cilindro 11.

- Durante la búsqueda, llegan solicitudes para los cilindros 1, 36, 16, 34, 9 y 12, en ese orden.
- La secuencia de búsqueda SSF será: 12, 9, 16, 1, 34, 36.
- Habrá un número de movimientos del brazo para un total de:
  - 111 cilindros según FCFS.
  - 61 cilindros según SSF.

El *algoritmo SSF* tiene el siguiente *problema*:

- El ingreso de nuevas solicitudes puede demorar la atención de las más antiguas.
- Con un disco muy cargado, el brazo tenderá a permanecer a la mitad del disco la mayoría del tiempo, como consecuencia de ello las solicitudes lejanas a la mitad del disco tendrán un mal servicio.
- Entran en conflicto los objetivos de:
  - Tiempo mínimo de respuesta.
  - Justicia en la atención.

La solución a este problema la brinda el **algoritmo del elevador** (por su analogía con el ascensor o elevador):

- Se mantiene el movimiento del brazo en la misma dirección, hasta que no tiene más solicitudes pendientes en esa dirección; entonces cambia de dirección.
- El software debe conservar el bit de dirección actual.

Ej. del *algoritmo del elevador* para el caso anterior, con el valor inicial arriba del bit de dirección:

- El orden de servicio a los cilindros es: 12, 16, 34, 36, 9 y 1.
- El número de movimientos del brazo corresponde a 60 cilindros.

El *algoritmo del elevador:*

- Ocasionalmente es mejor que el algoritmo SSF.
- Generalmente es peor que SSF.
- Dada cualquier colección de solicitudes, la cuota máxima del total de movimientos está fija, siendo el doble del número de cilindros.

Una variante consiste en *rastrear siempre en la misma dirección:*

- Luego de servir al cilindro con el número mayor:

  - El brazo pasa al cilindro de número menor con una solicitud pendiente.
  - Continúa su movimiento hacia arriba.

Algunos controladores de disco permiten que el software inspeccione el *número del sector activo debajo del cabezal:*

- Si dos o más solicitudes para el mismo cilindro están pendientes:
  - El manejador puede enviar una solicitud para el sector que pasará debajo del cabezal.
  - Se pueden hacer solicitudes consecutivas de distintas pistas de un mismo cilindro, sin generar un movimiento del brazo.

Cuando existen varias unidades, se debe tener una *tabla de solicitudes pendientes para cada unidad.*

Si una unidad está inactiva, deberá buscarse el cilindro siguiente necesario, si el controlador permite *búsquedas traslapadas.*

Cuando termina la transferencia actual se verifica si las unidades están en la posición del cilindro correcto:

- Si una o más unidades lo están, se puede iniciar la siguiente transferencia en una unidad ya posicionada.
- Si ninguno de los brazos está posicionado, el manejador:
  - Debe realizar una nueva búsqueda en la unidad que terminó la transferencia.
  - Debe esperar hasta la siguiente interrupción para ver cuál brazo se posiciona primero.

Generalmente, *las mejoras tecnológicas de los discos*:

- Acortan los *tiempos de búsqueda* (**seek**).
- No acortan los *tiempos de demora rotacional* (**search**).
- En algunos discos, el tiempo promedio de búsqueda ya es menor que el retraso rotacional.
- *El factor dominante será el retraso rotacional*, por lo tanto, los algoritmos que optimizan los tiempos de búsqueda (como el algoritmo del elevador) perderán importancia frente a los algoritmos que optimicen el retraso rotacional.

Una tecnología importante es la que permite el *trabajo conjunto de varios discos.*

Una configuración interesante es la de *treinta y ocho (38) unidades ejecutándose en paralelo.*

Cuando se realiza una operación de lectura:

- Ingresan a la cpu 38 bit a la vez, uno por cada unidad.

- Los 38 bits conforman una palabra de 32 bits junto con 6 bits para verificación.
- Los bits 1, 2, 4, 8, 16 y 32 se utilizan como *bits de paridad.*
- La palabra de 38 bits se puede codificar mediante el *código Hamming*, que es un *código corrector de errores.*
- Si una unidad sale de servicio:
  - Se pierde un bit de cada palabra.
  - El sistema puede continuar trabajando; se debe a que los códigos Hamming se pueden recuperar de un bit perdido.

Este diseño se conoce como **RAID;** siglas en inglés de "*arreglo redundante de discos no costosos*".

5.7 Porqué es Necesaria la Planificación de Discos

En los sistemas de multiprogramación muchos procesos pueden estar generando peticiones de e / s sobre discos [7, Deitel]:

- La generación de peticiones puede ser *mucho más rápida* que la atención de las mismas:
  - Se construyen *líneas de espera o colas* para cada dispositivo.
  - Para *reducir el tiempo de búsqueda* de registros *se ordena la cola de peticiones*: esto se denomina **planificación de disco**.

La planificación de disco implica:

- Un examen cuidadoso de las peticiones pendientes para determinar la *forma más eficiente* de servirlas.
- Un análisis de las *relaciones posicionales* entre las peticiones en espera.
- Un *reordenamiento de la cola de peticiones* para servirlas *minimizando los movimientos mecánicos.*

Los tipos más comunes de planificación son:

- Optimización de la *búsqueda.*
- Optimización *rotacional (latencia).*

Generalmente los tiempos de búsqueda superan a los de latencia, aunque la diferencia disminuye:

- Muchos algoritmos de planificación se concentran en la *reducción de los tiempos de búsqueda* para un conjunto de peticiones.

- Generalmente la *reducción de la latencia* recién tiene efectos bajo *cargas de trabajo muy pesadas.*

Bajo condiciones de *carga ligera* (promedio bajo de longitud de la cola), es aceptable el desempeño del método *FCFS* (primero en llegar, primero en ser servido).

Bajo condiciones de *carga media o pesada*, es recomendable un *algoritmo de planificación de las colas de requerimientos.*

5.8 Características Deseables de las Políticas de Planificación de Discos

Los principales *criterios de categorización de las políticas de planificación* son [7, Deitel]:

- *Capacidad de ejecución.*
- *Media del tiempo de respuesta.*
- *Varianza de los tiempos de respuesta* ***(predecibilidad).***

Una política de planificación debe intentar *maximizar la capacidad de ejecución*:

- Maximizar el número de peticiones servidas por unidad de tiempo.
- Minimizar la media del tiempo de respuesta.
- Mejorar el **rendimiento global**, quizás a costa de las peticiones individuales.

La planificación suele *mejorar la imagen total* al tiempo que reduce los niveles de servicio de ciertas peticiones:

- Se mide utilizando la *varianza de los tiempos de respuesta.*
- La varianza es un término estadístico que indica hasta qué punto tienden a desviarse del promedio de todos los elementos los elementos individuales.
- *A menor varianza mayor predecibilidad.*
- Se desea una política de planificación que minimice la varianza, es decir que *maximice la predecibilidad.*
- No debe haber peticiones que puedan experimentar niveles de servicio erráticos.

5.9 Optimización de la Búsqueda en Discos

Las *estrategias más comunes de optimización de la búsqueda* son las siguientes [7, Deitel]:

- *FCFS.*
- *SSTF.*

- *SCAN.*
- *SCAN de N - Pasos.*
- *C - SCAN.*
- *Esquema Eschenbach.*

5.9.1 Planificación FCFS (Primero en Llegar, Primero en Ser Servido)

Una petición no puede ser desplazada por la llegada de una petición con prioridad más alta.

No hay reordenamiento de la cola de peticiones pendientes.

Se ignoran las relaciones posicionales entre las peticiones pendientes.

Ofrece una *varianza pequeña* aunque perjudica a las peticiones situadas al final de la cola.

5.9.2 Planificación SSTF (Menor Tiempo de Búsqueda Primero)

El brazo del disco se sitúa en la siguiente petición que *minimice* el movimiento del brazo.

No respeta el orden de llegada de las peticiones a la cola.

Tiende a *favorecer a las pistas del centro* del disco.

La *media de tiempos de respuesta* tiende a ser *más baja que con FCFS*, para cargas moderadas.

Las *varianzas tienden a ser mayores que con FCFS* por el efecto de las pistas interiores y exteriores.

5.9.3 Planificación SCAN

El brazo del disco se desplaza sirviendo a todas las *peticiones que encuentra a su paso.*

Cambia de dirección cuando ya no hay peticiones pendientes en la dirección actual.

Ha sido la base de la mayoría de las estrategias de planificación implementadas.

Elimina las discriminaciones de *SSTF* y tiene menor varianza.

Las pistas exteriores son menos visitadas que las intermedias, pero no es tan grave como con *SSTF.*

5.9.4 Planificación SCAN de N - Pasos

La estrategia de movimiento del brazo es *como en SCAN*; solo da servicio a las peticiones que se encuentran en espera cuando comienza un recorrido particular.

Las peticiones que llegan *durante un recorrido* son agrupadas y ordenadas y serán atendidas durante el *recorrido de regreso.*

Posee menor varianza de los tiempos de respuesta si se compara con las planificaciones *SSTF* y *SCAN* convencionales.

5.9.5 Planificación C - SCAN (Búsqueda Circular)

El brazo se mueve del *cilindro exterior al interior*, sirviendo a las peticiones sobre una base de *búsqueda más corta.*

Finalizado el recorrido hacia el interior, salta a la petición más cercana al cilindro exterior y reanuda su desplazamiento hacia el interior.

No discrimina a los cilindros exterior e interior.

La *varianza de los tiempos de respuesta* es *muy pequeña.*

5.9.6 Esquema Eschenbach

El brazo del disco se mueve *como en C - SCAN*, pero:

- *Las peticiones se reordenan* para ser servidas dentro de un cilindro para tomar ventaja de la *posición rotacional.*
- Si dos peticiones trasladan posiciones de sectores dentro de un cilindro, solo se sirve una en el movimiento actual del brazo del disco.

Esta estrategia *tiene en cuenta el retraso rotacional.*

5.9.7 Conclusiones

Mediante trabajos de *simulación* y de *laboratorio* se demostró lo siguiente:

- *La estrategia SCAN es la mejor con carga baja.*
- *La estrategia C - SCAN es la mejor con cargas medias y pesadas.*
- *La estrategia C - SCAN con optimización rotacional es la mejor para cargas muy pesadas* (mejor que la estrategia Eschenbach inclusive).

5.10 Optimización Rotacional en Discos

En *condiciones de carga pesada*, las probabilidades de que ocurran referencias al mismo cilindro aumentan, por ello *resulta útil considerar la optimización rotacional además de la optimización de búsqueda [7, Deitel].*

La optimización rotacional es de uso común en dispositivos de cabezas fijas.

La estrategia utilizada es la *SLTF (tiempo de latencia más corto primero):*

- Situado el brazo del disco en un cilindro:
  - Examina todas las peticiones sobre el cilindro.
  - Sirve primero a la que tiene el *retraso rotacional más corto.*

5.11 Consideraciones de los Discos Sobre los Sistemas

Los *principales interrogantes* son [7, Deitel]:

- Cuándo es útil la planificación de disco.
- Cuándo puede degradar el rendimiento.

El almacenamiento en disco como un recurso limitador

La planificación de disco puede *mejorar el rendimiento y eliminar el embotellamiento*, que se produce cuando se concentran grandes cargas de peticiones sobre relativamente pocos discos o pocos cilindros de un disco.

Nivel de multiprogramación

Generalmente la planificación es efectiva en *sistemas de tiempo compartido* con un nivel alto de multiprogramación.

Subsistemas de discos múltiples

Frecuentemente la cpu está conectada mediante canales (o bus) a dispositivos controladores, los que están conectados a las unidades de discos.

El *embotellamiento* puede producirse en algún disco, algún controlador o en algún canal.

Existe *software específico* para:

- Medir la actividad.
- Detectar dónde se produce el embotellamiento.

Para eliminar ciertos embotellamientos puede ser necesaria una *reconfiguración del hardware*:

- Agregar canales, controladores, dispositivos.
- Cambiar dispositivos de un controlador a otro.
- Cambiar controladores de un canal a otro.

Para ayudar a *reducir la congestión del canal*, muchos sistemas han incorporado la *técnica de examen (sensado) de posición rotacional (RPS):*

- *Reduce el tiempo* durante el cual un canal se encuentra ocupado en la búsqueda de un registro.
- *RPS permite al canal quedar libre* justo hasta antes de que el registro se encuentre debajo de la cabeza de lectura - grabación apropiada.
- *RPS permite varias peticiones activas* al mismo tiempo en un solo canal, incrementando la performance.

Distribución de peticiones no uniformes

Son muy comunes en *ciertas situaciones reales.*

Son frecuentes en procesos secuenciales de archivos secuenciales, para los que se afectaron cilindros adyacentes inmediatos.

Generalmente en estos casos las búsquedas son cortas y la planificación de disco será de poca utilidad.

Técnicas de organización de archivos

Los *métodos de organización y acceso de archivos*, así como los *DBMS* (manejadores de bases de datos):

- Son muy convenientes desde el punto de vista de las aplicaciones y del usuario.
- Pueden generar complicaciones en la implementación y el rendimiento, puesto que el recorrido de estructuras de índices, bloques de control, apuntadores, etc., puede significar un gran número de operaciones de e / s.

5.12 Manejo de Errores en Discos

Algunos de los *errores más comunes* en discos son [23, Tanenbaum]:

- Error de programación:
  - Ej.: Solicitar un sector no existente.
- Error temporal en la suma de verificación:
  - Ej.: Provocado por polvo en la cabeza.
- Error permanente en la suma de verificación:
  - Ej.: Un bloque del disco dañado físicamente.
- Error de búsqueda:
  - Ej.: El brazo se envía al cilindro 6 pero va al 7.
- Error del controlador:
  - Ej.: El controlador no acepta los comandos.

El manejador del disco debe controlar los errores de la mejor manera posible.

La mayoría de los *controladores*:

- Verifican los parámetros que se les proporcionan.
- Informan si no son válidos.

Respecto de los *errores temporales* en la *suma de verificación*:

- Generalmente se eliminan al repetir la operación.
- Si persisten, el bloque debe ser marcado como un **bloque defectuoso**, para que el software lo evite.

Otra posibilidad es que *controladores "inteligentes"* reserven cierta cantidad de pistas:

- Serán asignadas en reemplazo de pistas defectuosas.
- Una **tabla** asocia las *pistas defectuosas* con las *pistas de repuesto*:
  - Está alojada en la memoria interna del controlador y en el disco.
  - La sustitución es transparente para el manejador.
  - Puede afectarse el desempeño de los algoritmos de búsqueda, como el del elevador, ya que el controlador utiliza pistas físicamente distintas de las solicitadas.

5.13 Ocultamiento de Una Pista a la Vez en Discos

Generalmente el tiempo de búsqueda supera al de rotación y transferencia (aunque esto se esta equilibrando) [23, Tanenbaum].

Una vez *resuelta la búsqueda del cilindro correspondiente, no es muy importante si se lee un sector o toda la pista*:

- Especialmente en dispositivos con *sensibilidad rotacional* (RPS):
  - El manejador puede ver que sector se encuentra debajo de la cabeza y puede enviar una solicitud del siguiente sector:
    - Permite *leer una pista en un tiempo de rotación.*
    - De lo contrario se tardaría, en promedio, un tiempo de rotación más un tiempo de sector, para leer un solo sector.
- Algunos manejadores aprovechan esto mediante un *caché secreto de una pista a la vez*:
  - Es *desconocido por el software* independiente del dispositivo.
  - Si se necesita un sector del caché, *no es necesaria una transferencia del disco.*
  - Las *principales desventajas* de este ocultamiento de una pista a la vez son:
    - Complejidad del software.
    - Requerimientos de espacio para buffers.
    - Las transferencias del caché al programa que hace la llamada:
      - Las debe realizar la cpu mediante un ciclo programado.
      - No las puede hacer el hardware DMA.
  - Algunos controladores realizan el ocultamiento de una pista a la vez en su propia memoria interna:
    - Resulta transparente al manejador.
    - Las transferencias entre el controlador y la memoria *pueden utilizar DMA*.

5.14 Discos en RAM

Utilizan una parte de la memoria principal asignada con anterioridad para almacenar los bloques [23, Tanenbaum].

Tienen la ventaja del *acceso instantáneo*:

- No hay demora rotacional o debida a las búsquedas.
- Son adecuados para el almacenamiento de programas o datos con accesos muy frecuentes.

Los bloques de almacenamiento tienen el mismo tamaño que en los discos reales.

Cuando el manejador debe leer de o escribir en un bloque de un disco en RAM, calcula el lugar de la memoria donde se encuentra el bloque solicitado y lee o escribe en el mismo.

5.15 Relojes

Los *relojes* o *cronómetros* son esenciales para la operación de sistemas de tiempo compartido [23, Tanenbaum].

Registran la hora del día.

Evitan que un proceso monopolice la cpu.

El *software para reloj* toma generalmente la forma de un *manejador de dispositivo*, aunque *no es un dispositivo de bloque ni de caracter*.

Los relojes más sencillo trabajan con la línea de corriente eléctrica de 110 o 220 voltios y provocan una interrupción por cada ciclo de voltaje, a 50 o 60 hz.

Otro tipo de relojes consta de tres componentes:

- Un oscilador de cristal, un contador y un registro.
- Una pieza de cristal de cuarzo se monta en una estructura bajo tensión:
  - Genera una señal periódica de muy alta precisión, generalmente entre 5 y 100 mhz.
  - La señal se alimenta en el contador para que cuente en forma descendente hasta cero.
  - Cuando el contador llega a cero, provoca una interrupción de la cpu.

Los *relojes programables* tienen varios modos de operación:

- *Modo de una instancia:*
  - Cuando el reloj se inicializa, copia el valor del registro en el contador.
  - Decrementa el contador en cada pulso del cristal.
  - Cuando el contador llega a cero provoca una interrupción y se detiene hasta ser nuevamente inicializado por el software.
- *Modo de onda cuadrada:*

- Luego de llegar a cero y provocar la interrupción, el registro se copia de manera automática en el contador.
- Todo el programa se repite en forma indefinida.
- Las interrupciones periódicas se llaman *marcas del reloj.*

La ventaja del *reloj programable* es que su *frecuencia de interrupción puede ser controlada por el software.*

Las principales *funciones del software manejador del reloj* son:

- Mantener la hora del día o *tiempo real.*
- Evitar que los procesos se ejecuten durante más tiempo del permitido.
- Mantener un registro del uso de la cpu.
- Controlar llamadas al sistema tipo "alarm" por parte de los procesos del usuario.
- Proporcionar cronómetros guardianes de partes del propio sistema.
- Realizar resúmenes, monitoreo y recolección de estadísticas.

El software manejador del reloj puede tener que *simular varios relojes virtuales con un único reloj físico.*

5.16 Terminales

Las terminales tienen gran número de formas distintas [23, Tanenbaum]:

- El *manejador de la terminal* debe ocultar estas diferencias.
- La parte *independiente del dispositivo* en el S. O. y los *programas del usuario* no se tienen que reescribir para cada tipo de terminal.

Desde el punto de vista del S. O. *se las puede clasificar en*:

- *Interfaz RS-232:*
  - Hardcopy (terminales de impresión).
  - TTY "de vidrio" (terminales de video).
  - Inteligente (computadoras con cpu y memoria).
- *Interfaz mapeada a memoria:*
  - Orientada a caracteres.
  - Orientada a bits.

Las *terminales RS-232* poseen un teclado y un monitor que se comunican mediante una *interfaz serial*, un bit a la vez; las conversiones de bits a bytes y viceversa las efectúan los chips uart (transmisores - receptores asíncronos universales).

Las *terminales mapeadas a memoria:*

- No se comunican mediante una línea serial.
- Poseen una interfaz mediante una memoria especial llamada *video RAM*:
  - Forma parte del espacio de direcciones de la computadora.
  - La cpu se dirige a ella como al resto de la memoria.
  - En la tarjeta de *video RAM* hay un chip llamado *controlador de video*:
    * Extrae bytes del video RAM y genera la señal de video utilizada para manejar la pantalla.
    * El monitor genera un rayo de electrones que recorre la pantalla pintando líneas.
    * Cada línea está constituida por un cierto número de puntos o *pixeles*.
    * La señal del controlador de video modula el rayo de electrones y determina si un pixel debe estar o no iluminado.
    * Los monitores de color poseen tres rayos (rojo, verde y azul) que se modulan independientemente.

En las *pantallas mapeadas a caracteres*:

- Cada caracter en la pantalla equivale a dos caracteres de RAM:
  - Uno aloja al código (ASCII) del caracter por exhibir.
  - Otro es el byte de atributo, necesario para determinar el color, el video inverso, el parpadeo, etc.

En las *terminales mapeadas a bits*:

- Se utiliza el mismo principio.
- Cada bit en el video RAM controla en forma directa un solo pixel de la pantalla.
- Permite una completa flexibilidad en los tipos y tamaños de caracteres, varias ventanas y gráficos arbitrarios.

Con las pantallas mapeadas a memoria, *el teclado se desacopla totalmente de la pantalla*:

- El teclado dispone de su propio manejador.
- El manejador del teclado puede operar en modo caracter o en modo línea.

Las terminales pueden operar con una *estructura central de buffers* o con *buffers exclusivos para cada terminal.*

Frecuentemente *los manejadores de terminales soportan operaciones* tales como:

- Mover el cursor hacia arriba, abajo, a la izquierda o a la derecha una posición.
- Mover el cursor a x,y.
- Insertar un caracter o una línea en el cursor.
- Eliminar un caracter o una línea en el cursor.
- Recorrer la pantalla hacia arriba o hacia abajo "n" líneas.
- Limpiar la pantalla desde el cursor hacia el final de la línea o hasta el final de la pantalla.
- Trabajar en modo de video inverso, subrayado, parpadeo o normal.
- Crear, construir, mover o controlar las ventanas.

Capítulo 6

Bloqueos

6.1 Introducción y Ejemplos de Bloqueo (o Interbloqueo)

Un proceso dentro de un sistema de multiprogramación está en un *estado de interbloqueo (o interbloqueado)* si está *esperando por un evento determinado que no ocurrirá [7, Deitel].*

Cuando los *recursos son compartidos* entre usuarios:

- Pueden producirse **interbloqueos** en los cuales los procesos de algunos usuarios nunca podrán llegar a su término.
- Se debe considerar la *prevención*, *evitación*, *detección* y *recuperación del interbloqueo* y la *postergación indefinida*, que se da cuando un proceso, aunque no esté interbloqueado, puede estar esperando por un *evento que probablemente nunca ocurrirá.*
- En algunos casos:
  - El precio de liberar interbloqueos en un sistema es demasiado alto.
  - Permitir el interbloqueo podría resultar catastrófico.

Los sistemas de cómputos tienen muchos *recursos que solo pueden ser utilizados por un proceso a la vez:*

- Ej.: impresoras, unidades de cinta, espacio de la tabla de *nodos-i.*
- Los S. O. tienen la capacidad de otorgar temporalmente a un proceso el *acceso exclusivo a ciertos recursos.*
- Frecuentemente un proceso necesita el **acceso exclusivo** no solo a un recurso, sino a varios.

Ej. de **bloqueo (deadlock):**

- Dos procesos desean imprimir grandes archivos en cinta.
- El proceso *"a"* solicita la impresora, que se le concede.
- El proceso *"b"* solicita la unidad de cinta, que se le concede.

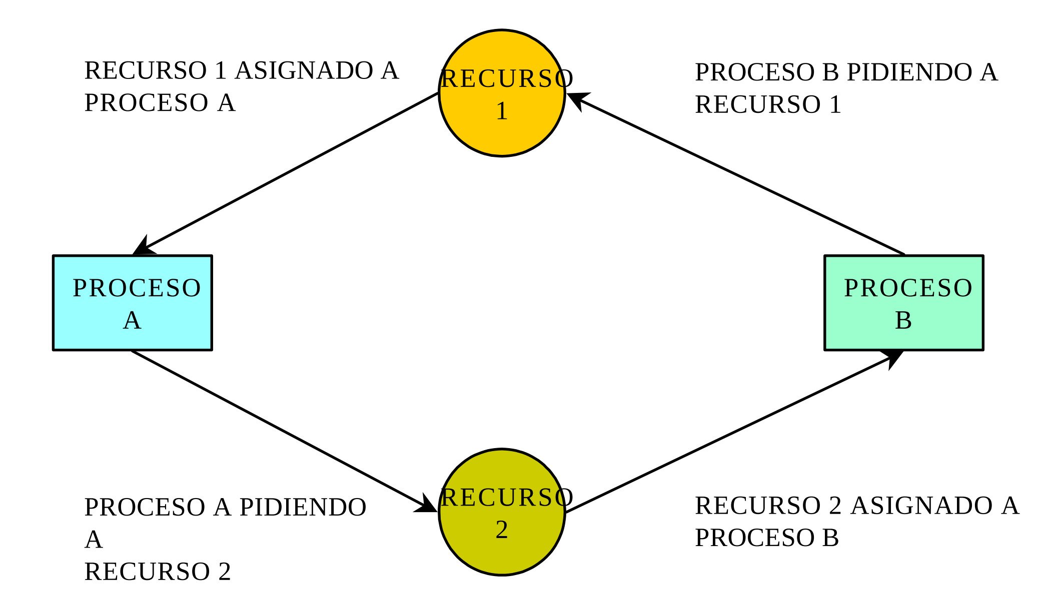


Figura 6.1: Un interbloqueo simple.

- El proceso *"a"* solicita la unidad de cinta, pero se deniega la solicitud hasta que *"b"* la libera.
- El proceso *"b"* solicita la impresora y se produce el *bloqueo (deadlock).*

Ejemplo de **interbloqueo de tráfico:**
Tiene similitud con el congestionamiento del tránsito en las ciudades.
El tráfico puede detenerse completamente.
Es necesaria una intervención externa para poner orden y restablecer la normalidad.
Ejemplo de **interbloqueo de un recurso simple:**
Tiene su origen en la contención normal de los recursos dedicados o reutilizables en serie:

- Pueden ser utilizados por un solo usuario a la vez.
- Cada proceso está esperando por el otro para liberar uno de los recursos.
- El recurso retenido no será liberado hasta que el otro proceso usuario libere su recurso.
- Este último proceso usuario no liberará su recurso retenido hasta que el primer proceso usuario libere su recurso retenido.
- Se produce una *espera circular.*[1]

Ejemplo de **interbloqueo en sistemas de spool:**
Un *sistema de spool* es utilizado para incrementar la capacidad de ejecución del sistema, al disasociar un programa de la lenta velocidad de los dispositivos (ej.: impresoras):

[1] Ver Figura 6.1 de la página 182 [7, Deitel].

- Si un programa envía líneas a una impresora, en realidad son enviadas a un dispositivo más rápido (disco).
- Se almacenan temporalmente hasta ser impresas.

Varios trabajos en ejecución que generan líneas de spool pueden interbloquearse si el espacio disponible se llena antes de completarse alguno de estos trabajos:

- Se *reduce la probabilidad de interbloqueos* del spool:
  - Proporcionando un espacio en disco considerablemente mayor que el necesario, preferentemente con asignación dinámica.
  - Limitando los spoolers de entrada para que no lean más trabajos cuando los archivos de spool llegan a cierto nivel de saturación.

Un problema relacionado: **postergación indefinida**:

Es posible que un proceso sea postergado indefinidamente en tanto que otros reciben la atención del sistema:

- Se trata de la *postergación indefinida.*
- Cuando los recursos son *planificados en función de prioridades*, un proceso dado puede esperar indefinidamente, mientras sigan llegando procesos de prioridades mayores.

En algunos sistemas, la postergación indefinida se evita al permitir que la prioridad de un proceso aumente mientras espera por un recurso; a esto se llama *envejecimiento.*

6.2 Conceptos de Recursos

El S. O. es, sobre todo, un *administrador de recursos [7, Deitel].*

Los recursos pueden ser "*apropiativos*", como la cpu y la memoria principal.

La apropiatividad es extremadamente importante para el éxito de los sistemas computacionales multiprogramados.

Ciertos recursos son "*no apropiativos*", como las unidades de cinta o cartridge magnéticos, o sea que no pueden sacarse de los procesos a los que están asignados.

Algunos recursos:

- Pueden ser **compartidos** entre varios procesos.
- Pueden estar **dedicados** a procesos individuales.

También son recursos compartibles (de uso compartido) ciertos programas:

- Se carga una copia del código a memoria.
- Se habilitan varias copias de las estructuras de datos, una para cada usuario.

- Como el código puede ser utilizado por varios usuarios a la vez, *no puede cambiar* durante la ejecución:
  - El *código que no cambia durante la ejecución* se denomina **reentrante**.
  - El *código que puede ser cambiado*, pero *se inicializa cada vez que se usa*, se denomina **reutilizable en serie**.

El código reentrante puede ser compartido simultáneamente por varios procesos.

El código reutilizable en serie puede ser usado solo por un proceso a la vez.

Cuando se consideran *compartidos* a determinados recursos, *se debe establecer si son utilizables por varios procesos simultáneamente o de a uno por vez*, estos últimos son los recursos que más a menudo están implicados en los interbloqueos.

6.3 Bloqueos y Condiciones Necesarias Para el Bloqueo

La *secuencia de eventos* necesarios para utilizar un recurso es la siguiente [23, Tanenbaum]:

1. **Solicitar** el recurso.
2. **Utilizar** el recurso.
3. **Liberar** el recurso.

Si el recurso *no está disponible* cuando se lo solicita:

- El proceso solicitante debe esperar.
- En algunos S. O. el proceso se bloquea automáticamente y se despierta cuando dicho recurso está disponible.
- En otros S. O. la solicitud falla y el proceso debe esperar para luego intentar nuevamente.

Un **bloqueo** se puede *definir formalmente* como sigue:

- *Un conjunto de procesos se bloquea si cada proceso del conjunto espera un evento que solo puede ser provocado por otro proceso del conjunto* [23, Tanenbaum]*:*
  - Ya que todos los procesos están esperando:
    - Ninguno realizará un evento que pueda despertar a los demás miembros del conjunto.
    - Todos los procesos esperarán por siempre.
  - Generalmente el evento que espera cada proceso es la liberación de cierto recurso que posee por el momento otro miembro del conjunto:
    - Cada miembro del conjunto de procesos bloqueados espera un recurso poseído por un proceso bloqueado.

  * Ninguno de los procesos bloqueados puede continuar su ejecución, ni liberar recursos, ni puede ser despertado.

Las *condiciones necesarias para el bloqueo* son (**Coffman**) [7, Deitel]:

- Los procesos reclaman control exclusivo de los recursos que piden (*condición de exclusión mutua*).
- Los procesos mantienen los recursos que ya les han sido asignados mientras esperan por recursos adicionales (*condición de espera por*).
- Los recursos no pueden ser extraídos de los procesos que los tienen hasta su completa utilización (*condición de no apropiatividad*).
- Existe una cadena circular de procesos en la que cada uno mantiene a uno o más recursos que son requeridos por el siguiente proceso de la cadena (*condición de espera circular*).

6.4 Modelación de Bloqueos

La *modelación de bloqueos* se puede mostrar mediante gráficas dirigidas (Holt).[2]

Las gráficas tienen dos tipos de nodos:

- Procesos (aparecen como círculos).
- Recursos (aparecen como cuadrados).
- Un arco de un nodo de recurso a uno de proceso indica que el recurso fue solicitado con anterioridad, fue otorgado y es poseído en ese momento por dicho proceso.
- Un arco de un proceso a un recurso indica que el proceso está bloqueado, en espera de ese recurso.
- Un ciclo en la gráfica indica la *existencia de un bloqueo* relacionado con los procesos y recursos en el ciclo.[3]

Las estrategias utilizadas para enfrentar los bloqueos son:

- Ignorar todo el problema.
- Detección y recuperación.
- Evitarlos dinámicamente mediante una cuidadosa asignación de recursos.
- Prevención mediante la negación estructural de una de las cuatro condiciones necesarias.

[2] Ver Figura 6.2 de la página 186 [23, Tanenbaum].

[3] Ver Figura 6.3 de la página 186 y Figura 6.4 de la página 187 [23, Tanenbaum].

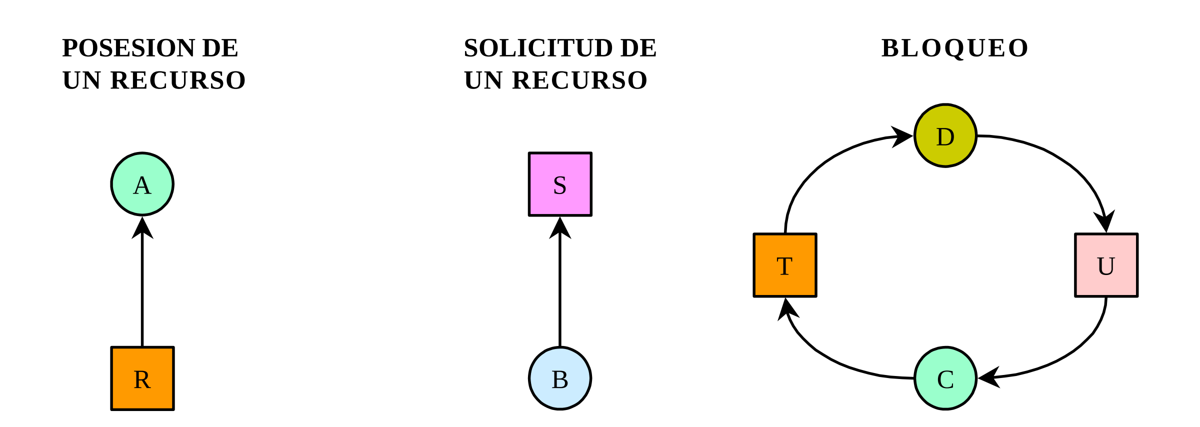


Figura 6.2: Gráficas de asignación de recursos.

PROCESOS: A, B, C RECURSOS: R, S, T

SECUENCIA DEL PROCESO A:
SOLICITUD DE R, SOLICITUD DE S, LIBERACION DE R, LIBERACION DE S.

SECUENCIA DEL PROCESO B:
SOLICITUD DE S, SOLICITUD DE T, LIBERACION DE S, LIBERACION DE T.

SECUENCIA DEL PROCESO C:
SOLICITUD DE T, SOLICITUD DE R, LIBERACION DE T, LIBERACION DE R.

SECUENCIA DE SOLICITUDES DE RECURSOS QUE CONDUCE A BLOQUEO:

A SOLICITUD R, B SOLICITUD S, C SOLICITUD T,
A SOLICITUD S, B SOLICITUD T, C SOLICITUD R, BLOQUEO.

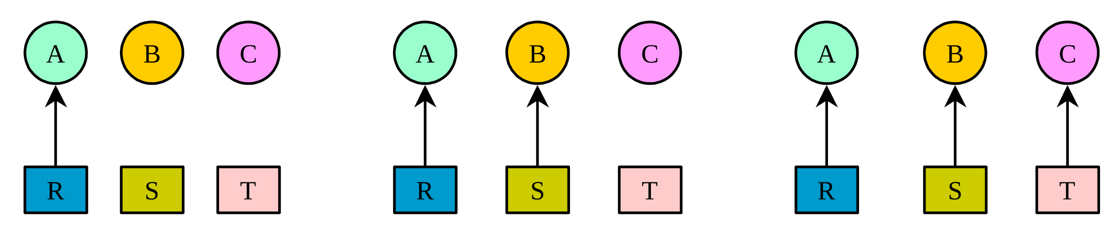
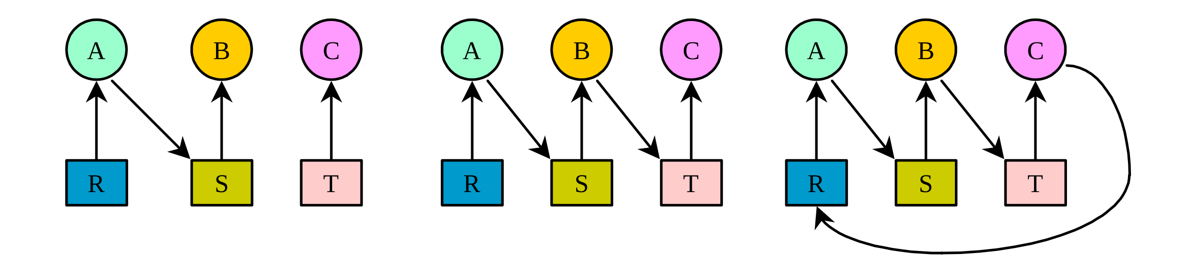


Figura 6.3: Ejemplo de ocurrencia de un bloqueo y la forma de evitarlo.



SECUENCIA DE SOLICITUDES DE RECURSOS QUE NO CONDUCE A BLOQUEO:

A SOLICITUD R, C SOLICITUD T, A SOLICITUD S,
C SOLICITUD R, A LIBERA R, A LIBERA S, NO EXISTE BLOQUEO.

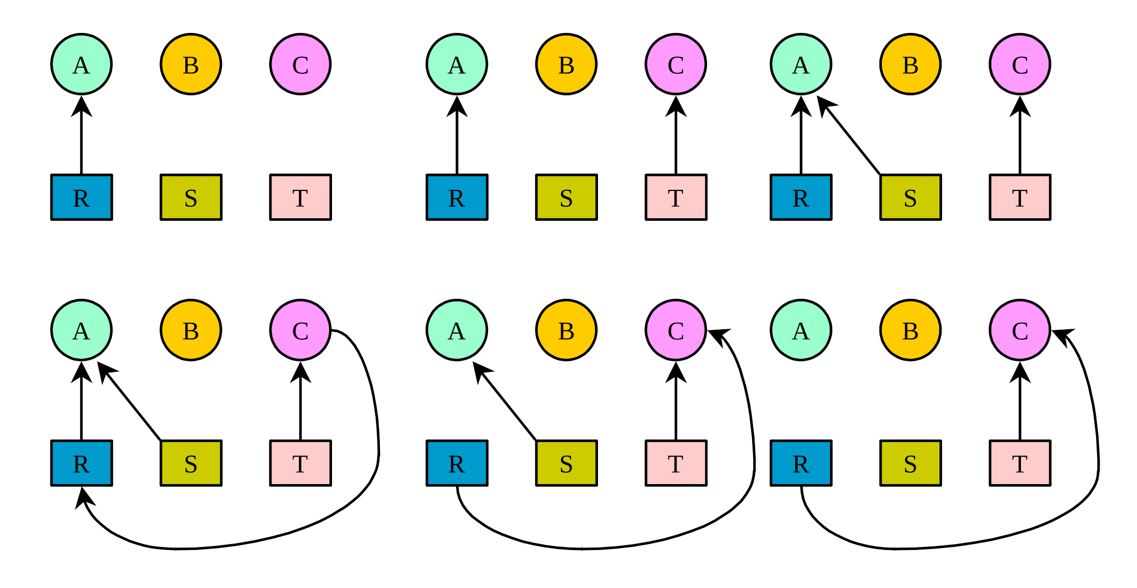


Figura 6.4: Ejemplo de ocurrencia de un bloqueo y la forma de evitarlo (continuación).

6.5 Areas Principales en la Investigación de Bloqueos

Los *principales aspectos* son los siguientes [7, Deitel]:

- *Prevención del bloqueo.*
- *Evitación del bloqueo.*
- *Detección del bloqueo.*
- *Recuperación del bloqueo.*

Prevención del bloqueo:

- El interés se centra en condicionar un sistema para que *elimine toda posibilidad* de que éstos se produzcan.
- Los métodos pueden dar como resultado una *pobre utilización de los recursos*, aún así son ampliamente utilizados.

Evitación del bloqueo:

- La meta es imponer *condiciones menos estrictas* que en la prevención, para intentar lograr una *mejor utilización de los recursos.*
- No precondiciona al sistema para que evite **todas** las posibilidades de que se produzca un bloqueo.
- Permiten la aparición del bloqueo, pero siempre que se produce una posibilidad de bloqueo, éste se *esquiva.*

Detección del bloqueo:

- Se utiliza en sistemas que *permiten que éstos ocurran*, ya sea voluntaria o involuntariamente.
- La meta es *determinar si ha ocurrido un bloqueo*:
  - Se debe **detectar** con precisión los procesos y recursos implicados en el bloqueo.
  - Se puede **eliminar** el bloqueo detectado.

Recuperación del bloqueo:

- Se utiliza para *despejar bloqueos* de un sistema para que:
  - Continúe operando sin ellos.
  - Terminen los procesos estancados.
  - Se liberen los recursos correspondientes a ellos.
- Generalmente se logra "*extrayendo*" *(cancelando)* a uno o varios de los procesos bloqueados, que se reinician luego de forma normal.

6.6 El Algoritmo del Avestrúz o de Ostrich

El punto de vista más simple es *pretender que no existe el problema* [23, Tanenbaum].

Esta estrategia puede generar distintas reacciones:

- Matemáticamente es inaceptable, considerándose que los bloqueos deben evitarse a toda costa.
- Desde la *ingeniería de software* podría considerarse cuál es la frecuencia esperada del problema, cuáles son sus consecuencias esperadas, cuáles son las frecuencias esperadas de fallas de otro tipo, etc.

Algunos S. O. soportan potencialmente bloqueos que ni siquiera se detectan, ya que se rompen automáticamente.

Los S. O. que ignoran el problema de los bloqueos *asumen la siguiente hipótesis:*

- La mayoría de los usuarios preferiría un bloqueo ocasional, en vez de una regla que restringiera a todos los usuarios en el uso de los distintos tipos de recursos.

El problema es que se debe pagar un cierto *precio para encarar el problema del bloqueo:*

- En restricciones para los procesos.
- En el uso de recursos.

Se presenta una contradicción entre la conveniencia y lo que es correcto.

Es muy difícil encontrar teóricamente soluciones prácticas de orden general aplicables a todos los tipos de S. O.

Un *criterio de orden general* utilizado por los S. O. que no hacen tratamiento específico del *bloqueo* consiste en:

- Intentar acceder al recurso compartido.
- De no ser factible el acceso:
  - Esperar un tiempo aleatorio.
  - Reintentar nuevamente.

6.7 Detección de Bloqueos

El S. O. no intenta evitar los bloqueos [23, Tanenbaum]*:*

- Intenta *detectar* cuando han ocurrido.
- Acciona para *recuperarse* después del hecho.

La detección del bloqueo es el proceso de:

- Determinar si de hecho existe o no un bloqueo.

- Identificar cuáles son los procesos y recursos implicados en el bloqueo.

Los algoritmos de detección de bloqueos implican cierta *sobrecarga en tiempo de ejecución*, por lo cual surge el siguiente interrogante: ¿ compensa la sobrecarga implícita en los algoritmos de detección de bloqueos, el ahorro potencial de localizarlos y romperlos ?.

6.7.1 Gráficas de Asignación de Recursos

Una *gráfica dirigida* indica las asignaciones y peticiones de recursos.

Los cuadros representan *procesos.*

Los círculos grandes indican clases de *recursos idénticos.*

Los círculos pequeños, dibujados dentro de los grandes, representan el *número de recursos idénticos dentro de cada clase.*[4]

6.7.2 Reducción de Gráficas de Asignación de Recursos

Si las peticiones de recursos de un proceso *pueden ser concedidas*, se dice que *una gráfica puede ser reducida por ese proceso.*

La *reducción de una gráfica* por un proceso determinado se muestra retirando:

- Las flechas que van de los recursos al proceso (los recursos asignados al proceso).
- Las flechas que van del proceso al recurso (las peticiones actuales del proceso).

Si una gráfica puede ser reducida por todos sus procesos, entonces no hay interbloqueo.

Si una gráfica no puede ser reducida por todos sus procesos, entonces los procesos "irreducibles" constituyen la serie de procesos interbloqueados de la gráfica.[5]

6.7.3 Detección de Bloqueos de Forma "Un Recurso de Cada Tipo"

No se dispone de más de *un objeto de cada clase de recurso.*

Si la gráfica de recursos contuviera uno o más ciclos, existiría un bloqueo.

Cualquier proceso que forme parte de un ciclo está bloqueado; si no existen ciclos, el sistema no está bloqueado.

Ejemplo: sistema con 7 (siete) procesos ("A" a "G") y 6 (seis) recursos ("R" a "W"):

- La *posesión* de los recursos es la siguiente:
  - El proceso A posee a R y desea a S.
  - El proceso B no posee recurso alguno y desea a T.
  - El proceso C no posee recurso alguno y desea a S.
  - El proceso D posee a U y desea a S y a T.
  - El proceso E posee a T y desea a V.
  - El proceso F posee a W y desea a S.

[4]Ver Figura 6.5 de la página 191 [7, Deitel].
[5]Ver Figura 6.6 de la página 192 [7, Deitel].

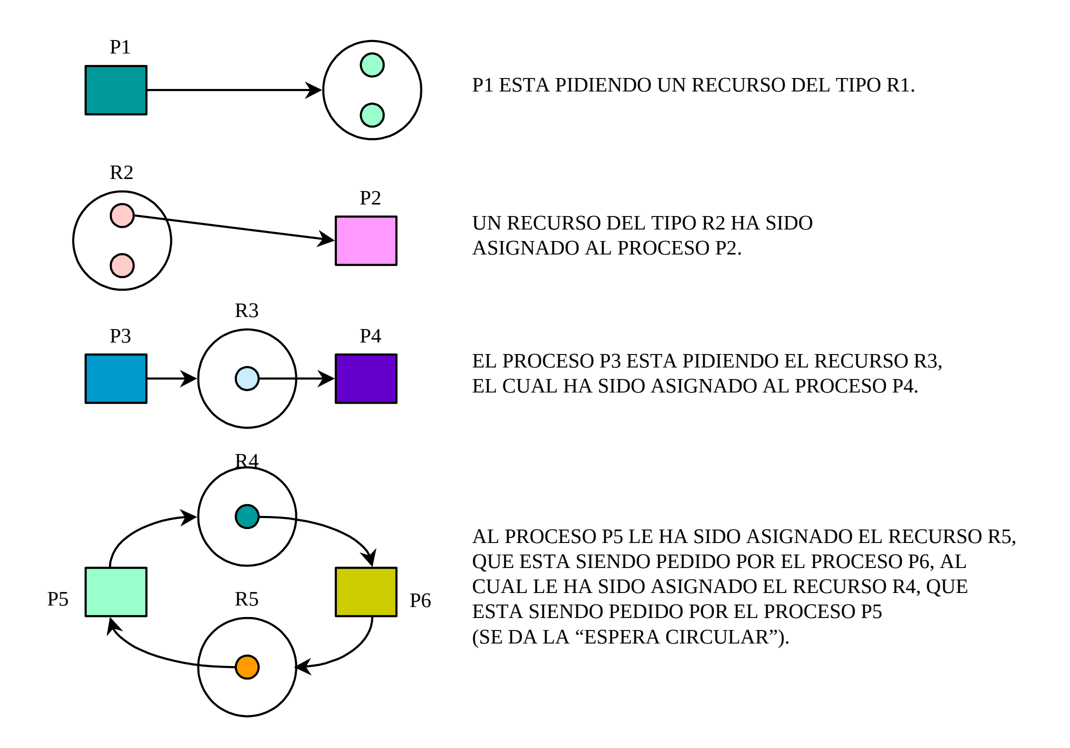


Figura 6.5: Gráfica de asignación y petición de recursos.

  - El proceso G posee a V y desea a U.

- La *pregunta* es: ¿está bloqueado este sistema y, en tal caso, cuáles son los procesos bloqueados?.

- La *respuesta* se obtiene mediante la **gráfica de recursos**: si la gráfica presenta un ciclo significa procesos bloqueados.[6]

Se hace necesario un *algoritmo formal* para la detección de bloqueos que se pueda utilizar en los *sistemas reales.*

Ejemplo de algoritmo aplicable a cada nodo “N” de la gráfica:

1. Se considera a “N” como nodo inicial.

2. Se inicializan:

   - La estructura de datos “L”como una lista vacía.
   - Todos los arcos como no marcados.

3. Se añade el nodo activo al final de “L” y se verifica si el nodo aparece en “L” dos veces:

   - Si aparece dos veces existe un ciclo y el algoritmo termina.

[6] Ver Figura 6.7 de la página 192 [23, Tanenbaum].

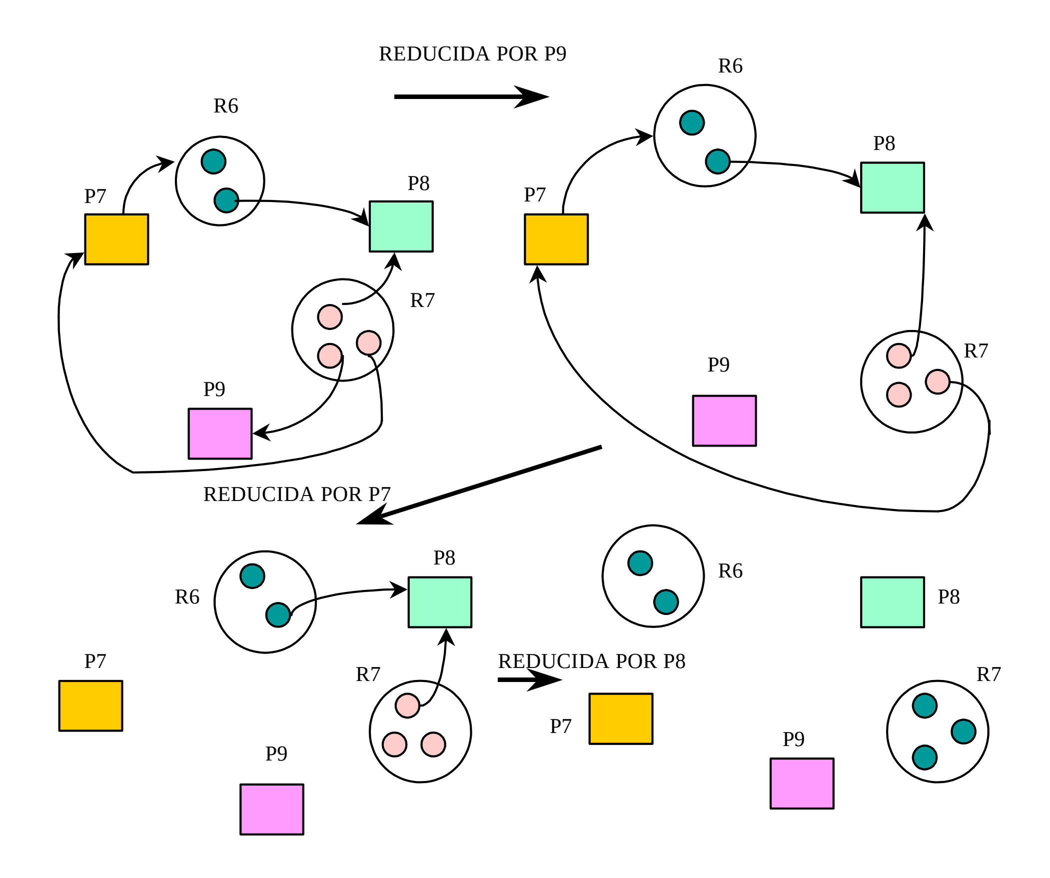


Figura 6.6: Reducciones de gráficas.

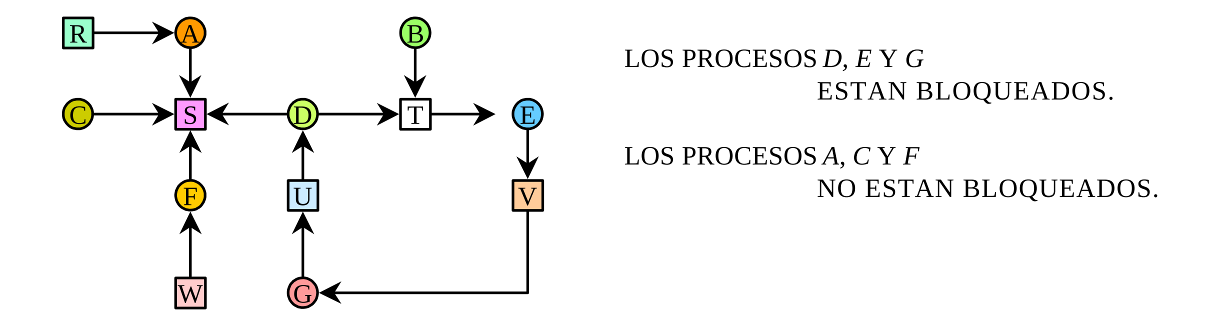


Figura 6.7: Gráfica de recursos y procesos.

4. Desde el nodo dado se verifica si existen arcos que salgan de dicho nodo y no estén marcados:
   - En caso afirmativo se va al paso 5.
   - En caso negativo se va al paso 6.
5. Se elige al azar un arco de salida no marcado y se le marca:
   - Luego se sigue este arco hasta el nuevo nodo activo y se regresa al paso 3.
6. Se ha llegado a un punto donde no se puede continuar:
   - Se regresa al nodo anterior, es decir al que estaba activo antes del actual.
   - Se señala de nuevo como nodo activo.
   - Se pasa al paso 3.
   - Si este nodo era el nodo inicial, la gráfica no contiene ciclos y el algoritmo termina.

La *aplicación del algoritmo precedente* al ejemplo anterior de gráfica dirigida es la siguiente:

- Se parte de "*R*" y se inicializa "*L*" como la lista vacía.
- Se añade "*R*" a la lista y se mueve a la única posibilidad, "*A*".
- Se añade "*A*" a la lista: *L=[R,A]*.
- Se pasa de "*A*" a "*S*", quedando *L=[R,A,S]*.
- "*S*" no tiene arcos que salgan de él, por lo que no se puede continuar y se regresa a "*A*".
- Ya que "*A*" no tiene arcos de salida no marcados se regresa a "*R*", finalizando la inspección de "*R*".
- Se inicia nuevamente el algoritmo partiendo de "*A*", siendo "*L*" otra vez la lista vacía.
- La búsqueda termina rápidamente y se parte de "*B*".
- De "*B*" se siguen los arcos de salida hasta llegar a "*D*", siendo *L=[B,T,E,V,G,U,D]*.
- Se efectúa una elección al azar.
- Si se elige "*S*" llegamos a un punto sin salida y debemos regresar a "*D*".
- La segunda vez se elige "*T*" quedando *L=[B,T,E,V,G,U,D,T]*:
  - Se ha descubierto un ciclo y el algoritmo se detiene.

6.7.4 Detección de Bloqueos de Forma "Varios Recursos de Cada Tipo"

Se considera un *algoritmo basado en matrices* para la detección de un bloqueo entre "n" *procesos*, "P_1" hasta "P_n".

Se considera "m" el número de *clases de recursos* con:

- E_1 recursos de la clase 1.
- E_2 recursos de la clase 2.
- E_i recursos de la clase "i" (1 menor o igual que "i" menor o igual que "m").
- *"E" es el vector de **recursos existentes**.*

En todo momento algunos de los recursos están asignados y por lo tanto no están disponibles.

Se considera un *vector "A" de **recursos disponibles**:*

- *"A_i"* indica el número de instancias disponibles del recurso "i" ; se refiere a *recursos no asignados.*

Se utilizan:

- La *matriz "C" de la **asignación actual**.*
- La *matriz "R" de **solicitudes**.*

El renglón i-ésimo de "C" indica el *número de instancias* de cada clase *"P_i"* poseídas en ese momento.

"C_{ij}" es el *número de instancias del recurso "j"* deseadas por "P_i".

Cada recurso está asignado o disponible, es decir que la *suma* de las instancias del recurso "j" *asignadas* y el número de instancias *disponibles* es el número de instancias *existentes* de esa clase de recurso [23, Tanenbaum].

Recursos en existencia *"E":*

$$\left(E_1, \quad E_2, \quad E_3, \quad .... \quad E_m \right)$$

Recursos disponibles *"A":*

$$\left(A_1, \quad A_2, \quad A_3, \quad .... \quad A_m \right)$$

Matriz de asignación actual *"C":*

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} & .... & C_{1m} \\ C_{21} & C_{22} & C_{23} & .... & C_{2m} \\ : & : & : & : & : \\ : & : & : & : & : \\ C_{n1} & C_{n2} & C_{n3} & .... & C_{nm} \end{bmatrix}$$

El renglón *"n"* es la asignación actual para el proceso *"n"*.

Matriz de solicitudes *"R":*

$$\begin{bmatrix} R_{11} & R_{12} & R_{13} & .... & R_{1m} \\ R_{21} & R_{22} & R_{23} & .... & R_{2m} \\ : & : & : & : & : \\ : & : & : & : & : \\ R_{n1} & R_{n2} & R_{n3} & .... & R_{nm} \end{bmatrix}$$

El renglón *"2"* es lo que necesita el proceso *"2"*.

Estructuras de datos necesarias para el algoritmo de detección de bloqueos.

El algoritmo de detección de bloqueos se basa en la comparación de vectores:

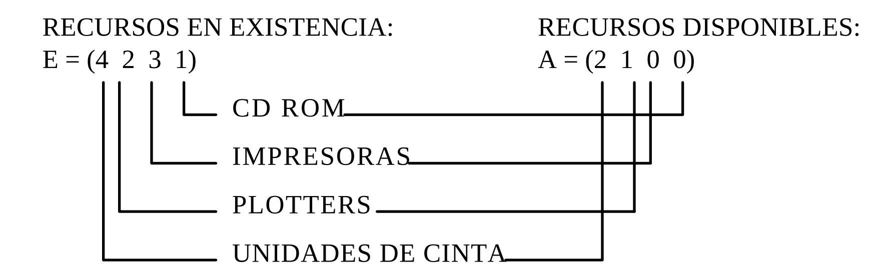
- Definimos que *"A"* es menor o igual que *"B"* si y solo si *"A_i"*es menor o igual que *"B_i"* para *"i"* entre *"0"* y *"m"*, ambos inclusive.

Los procesos no están marcados al principio.
Al avanzar el algoritmo los procesos se marcarán:

- Esto indica que pueden terminar su labor, ya que no están bloqueados.

- Al concluir el algoritmo se sabe que los procesos no marcados estarán bloqueados.

Los *pasos básicos del algoritmo de detección* de bloqueos son los siguientes:



MATRIZ DE ASIGNACION ACTUAL: MATRIZ DE SOLICITUDES:

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

Figura 6.8: Un ejemplo del algoritmo de detección de bloqueos.

1. Se busca un proceso no marcado *"P_i"* , para el cual el i-ésimo renglón de *"R"* sea menor que *"A"*.

2. Si se encuentra tal proceso, se suma el i-ésimo renglón de *"C"* a *"A"*, se marca el proceso y se regresa al paso 1.

3. Si no existe tal proceso, el algoritmo termina.

En el ejemplo tenemos 3 *procesos* y 4 *clases de recursos.*[7]
El proceso 1 tiene 1 impresora.
El proceso 2 tiene 2 unidades de cinta y 1 unidad de cd rom.
El proceso 3 tiene 1 plotter y 2 impresoras.
La matriz *"R"* indica las necesidades de *recursos adicionales*.
El *algoritmo de detección de bloqueos* busca un proceso cuya solicitud de un recurso pueda ser satisfecha:

- El proceso 1 no se puede satisfacer por no disponer de una unidad de cd rom.

- El proceso 2 no se puede satisfacer por no disponer de una impresora.

- El proceso 3 sí se puede satisfacer, por lo que se ejecuta, regresando en cierto momento sus recursos, lo que resulta en: A = (2 2 2 0).

Se ejecuta el proceso 2, el cual regresa sus recursos, obteniéndose: A = (4 2 2 1).
Se ejecuta el proceso restante: *no existe bloqueo* en el sistema.
Si se considera la siguiente variante:

- El proceso 2 necesita 1 unidad de cd rom, las 2 unidades de cinta y el plotter.

- No se pueden satisfacer las 3 solicitudes y *todo el sistema se bloquea.*

[7] Ver Figura 6.8 de la página 196 [23, Tanenbaum].

6.7.5 Cuándo Buscar los Bloqueos

Una posibilidad es *cada vez que se solicita un recurso*, pero esto podría sobrecargar al sistema.

Otra posibilidad es verificar *cada k minutos.*

Otro criterio es verificar *cuando el uso de la cpu baje de cierto valor fijo:*

- Si se bloquean suficientes procesos:
  - Existirán pocos procesos en ejecución.
  - La cpu estará inactiva con más frecuencia.

6.8 Recuperación de Bloqueos

Para *romper el bloqueo* de un sistema hay que *anular una o más de las condiciones necesarias para el bloqueo [7, Deitel].*

Normalmente, varios procesos *perderán* algo o todo lo realizado hasta el momento.

Los principales factores que dificultan la recuperación del bloqueo son los siguientes:

- Puede no estar claro si el sistema se ha bloqueado o no.
- Muchos sistemas tienen limitaciones para suspender un proceso por tiempo indefinido y reanudarlo más tarde:
  - Ej.: Los procesos de tiempo real, que deben funcionar continuamente, no son fáciles de suspender y reanudar.
- Los procedimientos de suspensión / reanudación implican una sobrecarga considerable.
- *La sobrecarga de recuperación está en función de la magnitud del bloqueo* (algunos, decenas o centenas de procesos involucrados).

Generalmente la **recuperación** suele realizarse:

- Retirando forzosamente (cancelando) a un proceso.
- Reclamando sus recursos.
- Permitiendo que los procesos restantes puedan finalizar.

Los procesos pueden ser retirados (cancelados) de acuerdo a un *orden de prioridades*, existiendo las siguientes dificultades:

- Pueden no existir las prioridades de los procesos bloqueados.
- Las prioridades instantáneas (en un momento dado), pueden ser incorrectas o confusas debido a consideraciones especiales, por ej.: procesos de baja prioridad que tienen prioridad alta momentáneamente debido a un tiempo tope inminente.

- La decisión óptima puede requerir un gran esfuerzo.

Algunas *formas de recuperación* ante bloqueos son [23, Tanenbaum]:

- *Recuperación mediante la apropiación.*
- *Recuperación mediante rollback.*
- *Recuperación mediante la eliminación de procesos.*

6.8.1 Recuperación Mediante la Apropiación

En ciertos casos podría ser posible *tomar un recurso temporalmente* de su poseedor y *dárselo a otro proceso,* por ej.:

- Retirar una impresora de un proceso para dedicarla a otro proceso.
- Retomar luego el primer proceso reasignándola al mismo.

La recuperación de recursos de esta forma depende en gran medida de la *naturaleza del recurso.*

La *elección del proceso a suspender* depende mucho:

- De cuáles procesos poseen recursos que pueden ser tomados con facilidad.
- De las posibilidades de recuperación luego de la apropiación.

6.8.2 Recuperación Mediante Rollback

En los S. O. donde es posible que ocurran bloqueos se puede hacer que *los procesos sean verificados periódicamente:*

- Su estado se graba en un archivo de modo que pueda volver a iniciar más tarde.
- El *punto de verificación* o de control contiene:
  - La imagen de la memoria.
  - El estado de los recursos, es decir, el detalle de los recursos asignados al proceso en ese instante.
- Los puntos de verificación grabados durante un proceso se mantienen sin ser regrabados.

Al detectarse un bloqueo es fácil ver cuáles son los recursos necesarios.

Para la **recuperación**, un proceso que posee un recurso necesario regresa hasta cierto instante en el tiempo anterior a la adquisición:

- Inicializa alguno de sus anteriores puntos de verificación.
- El proceso regresa a un momento anterior en el que no poseía el recurso.
- El recurso se asigna ahora a uno de los procesos bloqueados.
- Si el proceso que volvió a iniciar intenta adquirir de nuevo el recurso, tendrá que esperar hasta que esté disponible.

6.8.3 Recuperación Mediante la Eliminación de Procesos

- Es la *forma más sencilla* de romper un bloqueo.
- Una posibilidad es *eliminar un proceso del ciclo*: si el bloqueo no se rompe, se puede intentar con otro proceso del ciclo, hasta romper dicho ciclo.
- Otra posibilidad es *eliminar un proceso que no esté en el ciclo*, para poder liberar sus recursos: debe elegirse un proceso *que posea recursos necesarios por algún proceso del ciclo*.
- Siempre que sea posible, es mejor eliminar un proceso que pueda volver a iniciar su ejecución sin efectos dañinos:
  - Es preferible eliminar un proceso de compilación que un proceso de actualización de una base de datos:
    - La compilación se puede repetir sin problemas.
    - La actualización de una base de datos no siempre se puede repetir directamente.

6.9 Evasión de Bloqueos

En este análisis se supone implícitamente que si un proceso solicita recursos, los solicita *todos al mismo tiempo [23, Tanenbaum]:*

- En la mayoría de los sistemas los recursos se solicitan *uno a la vez.*
- El S. O. debe poder:
  - Decidir si el otorgamiento de un recurso es seguro o no.
  - Asignarlo solo en caso de que sea *seguro*.

El objetivo es *evitar el bloqueo haciendo la elección correcta* todo el tiempo, pero para evitar los bloqueos se requiere de cierta información de antemano.

6.9.1 Trayectorias de Recursos

Los principales algoritmos para *evitar los bloqueos* se basan en el concepto de *estados seguros.*[8]

El *ejemplo de modelo gráfico* utilizado indica lo siguiente:

- Es válido para dos procesos y dos recursos.
- El eje horizontal representa el número de instrucciones ejecutadas por el proceso *"A"*.
- El eje vertical representa el número de instrucciones ejecutadas por el proceso *"B"*.

[8] Ver Figura 6.9 de la página 200 [23, Tanenbaum].

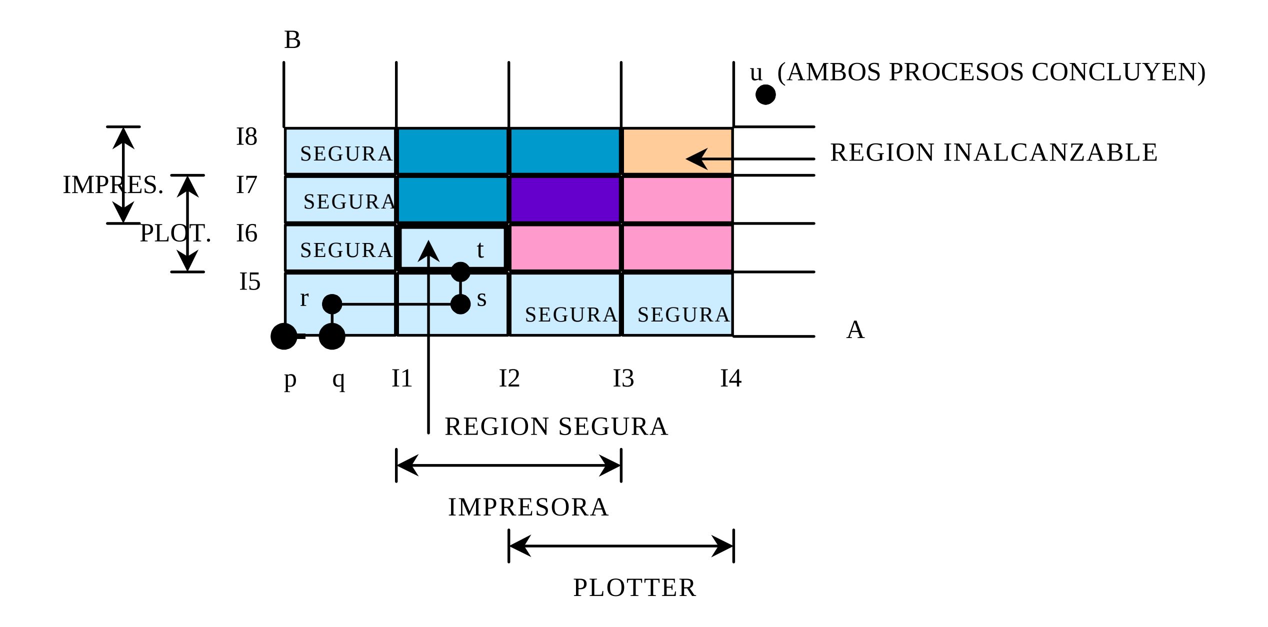


Figura 6.9: Trayectorias de recursos de dos procesos.

- En *"I_1"*, *"A"* solicita una impresora y en *"I_2"* necesita un plotter.
- En *"I_3"* e *"I_4"* se liberan la impresora y el plotter.
- El proceso *"B"* necesita el plotter desde *"I_5"* hasta *"I_7"* y la impresora desde *"I_6"* hasta *"I_8"*.
- Cada punto del diagrama representa un *estado conjunto* de los dos procesos.
- El estado inicial es *"p"*, sin que los procesos hayan ejecutado instrucción alguna.
- Si el planificador del S. O. elige *"A"* se pasa a *"q"*, en donde *"A"* ha ejecutado instrucciones pero no *"B"*.
- En *"q"* la trayectoria se vuelve vertical, ya que el planificador ha elegido ejecutar *"B"*.
- *Con un monoprocesador todas las trayectorias serán horizontales o verticales (no diagonales).*
- Cuando *"A"* cruza la línea *"I_1"* en la trayectoria de *"r"* a *"s"*, solicita y se le otorga la impresora.
- Cuando *"B"* alcanza el punto *"t"*, solicita el plotter.
- La región delimitada por *"I_1"*, *"I_3"*, *"I_6"* e *"I_8"* representa que *ambos* procesos poseen la impresora, pero esto es imposible y la *regla de exclusión mutua* impide la entrada a esta región.

- La región delimitada por *"I_2", "I_4", "I_5"* e *"I_7"* representa que *ambos* procesos poseen el plotter, lo que es imposible.

- Si el sistema ingresara a la región delimitada por *"I_1", "I_2", "I_5"* e *"I_6"* se bloqueará en la intersección de *"I_2"* e *"I_6":*

  - Acá, *"A"* solicita el plotter y *"B"* la impresora, que ya están asignados.
  - Toda *la región no es segura y no hay que entrar a ella*:
    - En *"t"*, lo único seguro es ejecutar *"A"* hasta llegar a *"I_4".*
    - Luego se puede utilizar cualquier trayectoria hasta *"u".*

- En *"t"*, *"B"* solicita un recurso:

  - El S. O. debe decidir si lo otorga o no.
  - Si lo otorga, el sistema entrará a una *región insegura* y se bloqueará en algún momento.
  - Para evitar el bloqueo, hay que suspender a *"B"* hasta que *"A"* haya solicitado y liberado el plotter.

6.9.2 Estados Seguros e Inseguros

Un *estado actual* está conformado por *"E", "A", "C"* y *"R":*

- *"E": vector de recursos en existencia.*

- *"A": vector de recursos disponibles.*

- *"C": matriz de asignación actual.*

- *"R": matriz de solicitudes.*

Un estado es **seguro** si:

- No está bloqueado.

- Existe una forma de satisfacer todas las solicitudes pendientes, mediante la ejecución de los procesos en cierto orden.

Ejemplo con un recurso para demostrar que *el estado en (a) es seguro*:

El estado es seguro ya que existe una sucesión de asignaciones que permiten terminar a todos los procesos; dicha sucesión de asignaciones es la siguiente:[9]

Ejemplo con un recurso para mostrar un *estado inseguro*:[10]

- No se puede garantizar que terminen los tres procesos.

- Si el proceso *"A"* pide y se le otorga una unidad, puede producirse un bloqueo de tres vías si cada uno de los procesos necesita al menos otra unidad del recurso antes de liberar ninguna.

(	a	)	(	b	)	(	c	)	(	d	)	(	e	)		
	T	M		T	M		T	M		T	M		T	M		
A	3	9	A	3	9	A	3	9	A	3	9	A	3	9	T:	Tiene
B	2	4	B	4	4	B	0	-	B	0	-	B	0	-	M:	Máximo
C	2	7	C	2	7	C	2	7	C	7	7	C	0	-	L:	Libre
	L:	3		L:	1		L:	5		L:	0		L:	7		

Tabla 6.1: Ejemplo de estado seguro en (a).

	T	M		
A	8	10	T:	Tiene
B	2	5	M:	Máximo
C	1	3	L:	Libre
	L:	1		

Tabla 6.2: Ejemplo de estado inseguro.

Un estado inseguro [7, Deitel]:

- *No implica la existencia, ni siquiera eventual, de bloqueo.*
- *Sí implica que alguna secuencia infortunada de eventos dé como resultado un bloqueo.*

La *diferencia entre estado seguro e inseguro* es que:

- A partir de un *estado seguro*, el sistema puede *garantizar la conclusión de todos los procesos.*
- A partir de un *estado inseguro*, *no existe tal garantía.*

Ejemplo de una *transición de estado seguro a estado inseguro*:[11]

- Dado un estado actual seguro, ello *no implica* que vayan a ser seguros todos los estados futuros.

6.9.3 El Algoritmo del Banquero (de Dijkstra) Para Solo Un Recurso

Es un *algoritmo de planificación* que puede *evitar los bloqueos [23, Tanenbaum].*

En la analogía:

- Los clientes son los *procesos*, las unidades de crédito son los *recursos* del sistema y el banquero es el *S. O.*

[9] Ver Tabla 6.1 de la página 202 [23, Tanenbaum].
[10] Ver Tabla 6.2 de la página 202 [7, Deitel].
[11] Ver Tabla 6.3 de la página 203 [7, Deitel].

	T	M		T	M		
A	1	4	A	1	4	T:	Tiene
B	4	6	B	4	6	M:	Máximo
C	5	8	C	6	8	L:	Libre
	L:	2		L:	1		

Tabla 6.3: Ejemplo de una transición de estado seguro a estado inseguro.

- El banquero sabe que no todos los clientes necesitaran su crédito máximo otorgado en forma inmediata, por ello reserva menos unidades (recursos) de las totales necesarias para dar servicio a los clientes.

Un estado inseguro no tiene que llevar a un bloqueo.
El algoritmo del banquero consiste en:

- Estudiar cada solicitud al ocurrir ésta.
- Ver si su otorgamiento conduce a un *estado seguro*:
  - En caso positivo, se otorga la solicitud.
  - En caso negativo, se la pospone.
- Para ver si un estado es seguro:
  - Verifica si tiene los recursos suficientes para satisfacer a otro cliente:
    - En caso afirmativo, se supone que los préstamos se pagarán.
    - Se verifica al siguiente cliente cercano al límite y así sucesivamente.
  - Si en cierto momento se vuelven a pagar todos los créditos, el estado es seguro y la solicitud original debe ser aprobada.

6.9.4 El Algoritmo del Banquero (de Dijkstra) Para Varios Recursos

Acá también los procesos deben establecer sus *necesidades totales de recursos antes de su ejecución* y dada una matriz de *recursos asignados*, el S. O. debe poder calcular en cualquier momento la matriz de *recursos necesarios.*[12]

Se dispone de:

- *"E": vector de recursos existentes.*
- *"P": vector de recursos poseídos.*
- *"A": vector de recursos disponibles.*

El algoritmo para determinar si un *estado es seguro* es el siguiente [23, Tanenbaum]:

[12] Ver Tabla 6.4 de la página 204 [23, Tanenbaum].

1. Se busca un renglón *"R"* cuyas necesidades de recursos no satisfechas sean menores o iguales que *"A":*
   - Si no existe tal renglón, el sistema se bloqueará en algún momento y ningún proceso podrá concluirse.
2. Supongamos que el proceso del renglón elegido solicita todos los recursos que necesita y concluye:
   - Se señala el proceso como concluido y se añaden sus recursos al vector *"A".*
3. Se repiten los pasos 1 y 2:
   - Hasta que todos los procesos queden señalados como concluidos, en cuyo caso, el estado inicial era seguro, o
   - Hasta que ocurra un bloqueo, en cuyo caso, no lo era.

Recursos Asignados					Recursos Necesarios					
A	3	0	1	1	A	1	1	0	0	
B	0	1	0	0	B	0	1	1	2	E=(6 3 4 2)
C	1	1	1	0	C	3	1	0	0	P=(5 3 2 2)
D	1	1	0	1	D	0	0	1	0	A=(1 0 2 0)
E	0	0	0	0	E	2	1	1	0	
↑										
Procesos										

Tabla 6.4: El algoritmo del banquero con varios recursos.

6.9.5 Asignación de Recursos por el Algoritmo del Banquero

Se permiten las condiciones de *"exclusión mutua", "espera por"* y *"no apropiatividad" [7, Deitel].*

Los procesos reclaman uso exclusivo de los recursos que requieren.

Los procesos mantienen los recursos mientras piden y esperan por otros recursos adicionales, pero no pueden apropiarse de un proceso que mantenga esos recursos.

Las peticiones son de un recurso a la vez.

El S. O. puede conceder o negar cada una de las peticiones; si se niega una petición:

- El proceso retiene los recursos que ya tiene asignados.
- Espera un tiempo finito hasta que le sea atendida la petición.

El S. O. concede peticiones que den como resultado solo estados seguros.

Dado que el sistema se mantiene siempre en *estado seguro*, todas las peticiones serán atendidas en un tiempo finito.

6.9.6 Debilidades del Algoritmo del Banquero

Requiere que exista un *número fijo de recursos asignables*, pero generalmente no se puede contar con que el número de recursos se mantenga siempre constante [7, Deitel].

Requiere que la *población de usuarios se mantenga constante*, lo cual es irrazonable.

Requiere que el S. O. garantice que *todas las peticiones serán concedidas en un tiempo finito*, pero en la realidad se requieren mayores garantías.

Requiere que los procesos *reintegren los recursos en un tiempo finito*, pero en la realidad se requieren mayores garantías.

Requiere que los procesos indiquen sus *necesidades máximas de recursos por adelantado*, lo cual generalmente no ocurre.

Generalmente no es utilizado en S. O. reales.

6.10 Prevención de Bloqueos

Si se puede garantizar que *al menos una* de las cuatro *condiciones de Coffman* para el bloqueo nunca se satisface, entonces los bloqueos serán imposibles por razones estructurales *(enunciado de Havender) [23, Tanenbaum].*

Havender sugirió las siguientes *estrategias para evitar varias de las condiciones de bloqueo:*

- Cada proceso [7, Deitel]:
  - Deberá pedir todos sus recursos requeridos de una sola vez.
  - No podrá proceder hasta que le hayan sido asignados.
- Si a un proceso que mantiene ciertos recursos se le niega una nueva petición, este proceso deberá:
  - Liberar sus recursos originales.
  - En caso necesario, pedirlos de nuevo junto con los recursos adicionales.
- Se impondrá la ordenación lineal de los tipos de recursos en todos los procesos:
  - Si a un proceso le han sido asignados recursos de un tipo dado, en lo sucesivo solo podrá pedir aquellos recursos de los tipos que siguen en el ordenamiento.

Havender no presenta una estrategia contra el uso exclusivo de recursos por parte de los procesos, pues se desea permitir el uso de recursos dedicados.

6.10.1 Prevención de la Condición de Exclusión Mutua

Si ningún recurso se asignara de manera exclusiva a un solo proceso, nunca tendríamos bloqueos, pero esto es imposible de aplicar, en especial en relación a ciertos tipos de recursos, que en un momento dado no pueden ser compartidos (ej.: impresoras).

Se debe:

- Evitar la asignación de un recurso cuando no sea absolutamente necesario.
- Intentar asegurarse de que los menos procesos posibles puedan pedir el recurso.

6.10.2 Prevención de la Condición "detenerse y esperar" o "espera por"

Si se puede *evitar que los procesos que conservan recursos esperen más recursos*, se pueden eliminar los bloqueos.

Una forma es exigir a *todos los procesos que soliciten todos los recursos* antes de iniciar su ejecución; si un proceso no puede disponer de todos los recursos, deberá esperar, pero sin retener recursos afectados.

Un problema es que muchos procesos no *saben* el número de recursos necesarios hasta iniciar su ejecución.

Otro problema es que puede significar desperdicio de recursos, dado que todos los recursos necesarios para un proceso están afectados al mismo desde su inicio hasta su finalización.

Otro criterio aplicable consiste en:

- Exigir a un proceso que solicita un recurso que libere en forma temporal los demás recursos que mantiene en ese momento.
- Hacer que el proceso intente luego recuperar todo al mismo tiempo.

6.10.3 Prevención de la Condición de "no apropiación"

Una de las estrategias de Havender requiere que cuando a un proceso que mantiene recursos le es negada una petición de recursos adicionales; deberá liberar sus recursos y si es necesario pedirlos de nuevo junto con los recursos adicionales.

La implementación de esta estrategia *niega la condición de "no apropiación"* y *los recursos pueden ser retirados* de los procesos que los retienen *antes de la terminación de los procesos.*

El problema consiste en que el retiro de ciertos recursos de un proceso *puede significar:*

- La pérdida del trabajo efectuado hasta ese punto.
- La necesidad de repetirlo luego.

Una consecuencia seria es la *posible postergación indefinida* de un proceso.

6.10.4 Prevención de la Condición de "espera circular"

Una forma es que un proceso solo está autorizado a *utilizar un recurso en cada momento:*

- Si necesita otro recursos, debe liberar el primero.
- Esto resulta inaceptable para muchos procesos.

Otra forma es la siguiente:

- Todos los recursos se numeran globalmente.
- Los procesos pueden solicitar los recursos en cualquier momento:

   - Las solicitudes se deben hacer según un cierto orden numérico (creciente) de recurso; debido a lo cual la gráfica de asignación de recursos no tendrá ciclos.

- En cada instante uno de los recursos asignados tendrá el número más grande:

   - El proceso que lo posea no pedirá un recurso ya asignado.
   - El proceso terminará o solicitará recursos con números mayores , que estarán disponibles:
      * Al concluir liberará sus recursos.
      * Otro proceso tendrá el recurso con el número mayor y también podrá terminar.
      * Todos los procesos podrán terminar y no habrá bloqueo.

Una *variante* consiste en *eliminar el requisito de adquisición de recursos en orden creciente*:

- Ningún proceso debe solicitar un recurso con número menor al que posee en el momento.

El problema es que en casos reales podría resultar imposible encontrar un orden que satisfaga a todos los procesos.

6.11 Otros Aspectos

Los *métodos para prevenir el bloqueo* pueden resumirse según se indica en la Tabla 6.5 de la página 207 [23, Tanenbaum].

Condición	Método
Exclusión mutua	Realizar un spooling general
Detenerse y esperar	Solicitar todos los recursos al principio
No apropiación	Retirar los recursos
Espera circular	Ordenar los recursos en forma numérica

Tabla 6.5: Resumen de los métodos para prevenir el bloqueo.

Otros aspectos interesantes relacionados con bloqueos son [23, Tanenbaum]:

- *La cerradura de dos fases.*

- *Los bloqueos sin recursos.*

- *La inanición.*

6.11.1 Cerradura de Dos Fases

Una operación frecuente en sistemas de bases de datos consiste en:

- Solicitar el cierre de varios registros.
- Actualizar todos los registros cerrados.
- Ante la ejecución de varios procesos al mismo tiempo, existe un grave riesgo de bloqueo.

El método de la *cerradura de dos fases* consiste en:

- **Primer fase**: el proceso intenta cerrar todos los registros necesarios, uno a la vez.
- **Segunda fase**: se actualiza y se liberan las cerraduras.
- Si durante la primer fase se necesita algún registro ya cerrado:
  - El proceso libera todas las cerraduras y comienza en la primer fase nuevamente.
  - Generalmente esto no resulta aplicable en la realidad:
    - No resulta aceptable dejar un proceso a la mitad y volver a comenzar.
    - El proceso podría haber actualizado archivos, enviado mensajes en la red, etc.

6.11.2 Bloqueos Sin Recursos

Los bloqueos también pueden aparecer en *situaciones que no están relacionadas con los recursos.*

Puede ocurrir que dos procesos se bloqueen en espera de que el otro realice cierta acción, por ej.: operaciones efectuadas sobre semáforos (indicadores o variables de control) en orden incorrecto.

6.11.3 Inanición

En un sistema dinámico permanentemente hay solicitudes de recursos.

Se necesita un criterio (política) para decidir:

- *Quién* obtiene cual recurso.
- En *qué* momento.

Podría suceder que ciertos procesos *nunca lograran el servicio*, aún sin estar bloqueados, porque se privilegia en el uso del recurso a otros procesos.

La **inanición** *se puede evitar mediante el criterio de asignación de recursos FIFO* "el primero en llegar es el primero en despachar (ser atendido)".

El proceso que ha esperado el máximo tiempo se despachará a continuación:

- En el transcurso del tiempo, cualquiera de los procesos dados:
  - Será el más antiguo.
  - Obtendrá el recurso necesario.

6.12 Tendencias del Tratamiento del Bloqueo

Generalmente los S. O. han considerado al bloqueo como una *incomodidad limitada [7, Deitel]*.

Muchos S. O. implementan métodos básicos de prevención de bloqueos sugeridos por Havender y los resultados son satisfactorios en gran número de casos.

La tendencia es a que el bloqueo *tenga una consideración mucho mayor en los nuevos S. O.*, debido a:

- Orientación hacia la operación asincrónica en paralelo:
  - Incremento del multiprocesamiento y de las operaciones concurrentes.
- Asignación dinámica de recursos:
  - Capacidad de los procesos de adquirir y liberar recursos según las necesidades.
  - Ignorancia a priori de los procesos respecto de sus necesidades de recursos.
- Consideración de los datos como un recurso:
  - Significa incrementar la capacidad del S. O. para administrar gran número de recursos.

Parte II

Sistemas Operativos Distribuidos

Capítulo 7

Introducción a los Sistemas Distribuidos

7.1 Introducción a los Sistemas Distribuidos

Desde el *inicio de la era de la computadora moderna* (1945), hasta cerca de 1985, solo se conocía la *computación centralizada [25, Tanenbaum].*

A partir de la *mitad de la década de los ochentas* aparecen dos avances tecnológicos fundamentales:

- Desarrollo de **microprocesadores** poderosos y económicos con arquitecturas de 8, 16, 32 y 64 bits.
- Desarrollo de **redes de área local** (*LAN*) de alta velocidad, con posibilidad de conectar cientos de máquinas a velocidades de transferencia de millones de bits por segundo (mb/seg).

Aparecen los **sistemas distribuidos**, en contraste con los *sistemas centralizados.*

Los sistemas distribuidos necesitan un software distinto al de los sistemas centralizados.

Los S. O. para sistemas distribuidos han tenido importantes desarrollos pero todavía existe un largo camino por recorrer.

Los usuarios pueden acceder a una *gran variedad de recursos computacionales:*

- De hardware y de software.
- Distribuidos entre un gran número de sistemas computacionales conectados.

Un importante antecedente de las redes de computadoras lo constituye Arpanet, iniciada en 1968 en los EE. UU.

7.2 Ventajas de los Sistemas Distribuidos con Respecto a los Centralizados

Una razón para la tendencia hacia la descentralización es la *economía.*

Herb Grosch formuló la que se llamaría *"Ley de Grosch" [25, Tanenbaum]:*

- El poder de cómputo de una cpu es proporcional al cuadrado de su precio:
  - Si se paga el doble se obtiene el cuádruple del desempeño.
- Fue aplicable en los años setentas y ochentas a la *tecnología mainframe*.
- No es aplicable a la tecnología del microprocesador:
  - La solución más eficaz en cuanto a costo es limitarse a un gran número de cpu baratos reunidos en un mismo sistema.

Los sistemas distribuidos generalmente tienen en potencia una proporción precio / desempeño mucho mejor que la de un único sistema centralizado.

Algunos autores distinguen entre:

- **Sistemas distribuidos**: están diseñados para que *muchos usuarios trabajen en forma conjunta.*
- **Sistemas paralelos**: están diseñados para lograr la *máxima rapidez en un único problema.*

En general se consideran *sistemas distribuidos, en sentido amplio*, a los sistemas en que:

- Existen *varias cpu conectadas* entre sí.
- Las distintas cpu *trabajan de manera conjunta.*

Ciertas *aplicaciones* son distribuidas en forma inherente:

- Ej.: sistema de automatización de una fábrica:
  - Controla los robots y máquinas en la línea de montaje.
  - Cada robot o máquina es controlado por su propia computadora.
  - Las distintas computadoras están interconectadas.

Una *ventaja potencial de un sistema distribuido* es una mayor *confiabilidad:*

- Al distribuir la carga de trabajo en muchas máquinas, la falla de una de ellas no afectara a las demás:
  - La carga de trabajo podría distribuirse.
- Si una máquina se descompone:
  - Sobrevive el sistema como un todo.

Otra ventaja importante es la posibilidad del *crecimiento incremental* o por incrementos:

- Podrían añadirse procesadores al sistema, permitiendo un desarrollo gradual según las necesidades.
- No son necesarios grandes incrementos de potencia en breves lapsos de tiempo.
- Se puede añadir poder de cómputo en pequeños incrementos.

7.3 Ventajas de los Sistemas Distribuidos con Respecto a las PC Independientes

Satisfacen la necesidad de muchos usuarios de *compartir ciertos datos [25, Tanenbaum]*:

- Ej.: sistema de reservas de líneas aéreas.

También con los sistemas distribuidos se pueden *compartir otros recursos como programas y periféricos costosos:*

- Ej.: impresoras láser color, equipos de fotocomposición, dispositivos de almacenamiento masivo (ej.: cajas ópticas), etc.

Otra importante razón es lograr una *mejor comunicación entre las personas:*

- Ej.: correo electrónico:
  - Posee importantes ventajas sobre el correo por cartas, el teléfono y el fax:
    - Velocidad, disponibilidad, generación de documentos editables por procesadores de texto, etc.

La *mayor flexibilidad* es también importante:

- La carga de trabajo se puede difundir (distribuir) entre las máquinas disponibles en la forma más eficaz según el criterio adoptado (por ej. costos).
- Los equipos distribuidos pueden no ser siempre PC:
  - Se pueden estructurar sistemas con grupos de PC y de computadoras compartidas, de distinta capacidad.

7.4 Desventajas de los Sistemas Distribuidos

El principal problema es el software, ya que el diseño, implantación y uso del software distribuido presenta numerosos inconvenientes [25, Tanenbaum].

Los *principales interrogantes* son los siguientes:

- ¿Qué tipo de S. O., lenguaje de programación y aplicaciones son adecuados para estos sistemas?.
- ¿Cuánto deben saber los usuarios de la distribución?.

- ¿Qué tanto debe hacer el sistema y qué tanto deben hacer los usuarios?.

La *respuesta a estos interrogantes no es uniforme* entre los especialistas, pues existe una gran *diversidad de criterios* y de *interpretaciones* al respecto.

Otro problema potencial tiene que ver con las *redes de comunicaciones*, ya que se deben considerar problemas debidos a pérdidas de mensajes, saturación en el tráfico, expansión, etc.

El hecho de que sea fácil *compartir los datos* es una ventaja pero se puede convertir en un gran problema, por lo que la seguridad debe organizarse adecuadamente.

En general se considera que las ventajas superan a las desventajas, si estas últimas se administran seriamente.

7.5 Conceptos de Hardware

Todos los sistemas distribuidos constan de varias cpu, organizadas de diversas formas, especialmente respecto de [25, Tanenbaum]:

- La *forma de interconectarlas* entre sí.
- Los *esquemas de comunicación* utilizados.

Existen diversos *esquemas de clasificación* para los sistemas de cómputos con varias cpu:

- Uno de los mas conocidos es la *"Taxonomía de Flynn":*
  - Considera como características esenciales el *número de flujo de instrucciones* y el *número de flujos de datos.*
  - La clasificación incluye equipos **SISD, SIMD, MISD** y **MIMD.**

SISD (Single Instruction Single Data: un flujo de instrucciones y un flujo de datos):

- Poseen un único procesador.

SIMD (Single Instruction Multiple Data: un flujo de instrucciones y varios flujos de datos):

- Se refiere a ordenar procesadores con una unidad de instrucción que:
  - Busca una instrucción.
  - Instruye a varias unidades de datos para que la lleven a cabo en paralelo, cada una con sus propios datos.
- Son útiles para los cómputos que repiten los mismos cálculos en varios conjuntos de datos.

MISD (Multiple Instruction Single Data: un flujo de varias instrucciones y un solo flujo de datos):

- No se presenta en la práctica.

MIMD (Multiple Instruction Multiple Data: un grupo de computadoras independientes, cada una con su propio contador del programa, programa y datos):

- **Todos los sistemas distribuidos son de este tipo.**

Un avance sobre la clasificación de Flynn incluye la división de las computadoras *MIMD* en dos grupos:

- *Multiprocesadores*: poseen memoria compartida:
  - Los distintos procesadores comparten el mismo espacio de direcciones virtuales.
- *Multicomputadoras*: no poseen memoria compartida:
  - Ej.: grupo de PC conectadas mediante una red.

Cada una de las categorías indicadas se puede *clasificar según la arquitectura de la red de interconexión* en:

- *Esquema de bus:*
  - Existe una sola red, bus, cable u otro medio que conecta *todas* las máquinas:
    * Ej.: la televisión por cable.
- *Esquema con conmutador:*
  - No existe una sola columna vertebral de conexión:
    * Hay múltiples conexiones y varios patrones de conexionado.
    * Los mensajes de mueven a través de los medios de conexión.
    * Se decide explícitamente la conmutación en cada etapa para dirigir el mensaje a lo largo de uno de los cables de salida.
    * Ej.: el sistema mundial telefónico público.

Otro aspecto de la clasificación considera el *acoplamiento entre los equipos:*

- *Sistemas fuertemente acoplados:*
  - El retraso al enviar un mensaje de una computadora a otra es corto y la tasa de transmisión es alta.
  - Generalmente se los utiliza como sistemas paralelos.
- *Sistemas débilmente acoplados:*
  - El retraso de los mensajes entre las máquinas es grande y la tasa de transmisión es baja.
  - Generalmente se los utiliza como sistemas distribuidos.

Generalmente los multiprocesadores están más fuertemente acoplados que las multicomputadoras.

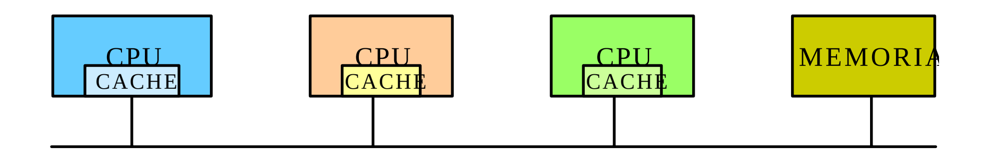


Figura 7.1: Multiprocesadores con base en un bus.

7.6 Multiprocesadores con Base en Buses

Constan de cierto número de cpu conectadas a un *bus común*, junto con un módulo de memoria.[1]

Un bus típico posee al menos [25, Tanenbaum]:

- 32 líneas de direcciones.
- 32 líneas de datos.
- 30 líneas de control.

Todos los elementos precedentes *operan en paralelo.*

Para *leer* una palabra de memoria, una cpu:

- Coloca la dirección de la palabra deseada en las líneas de direcciones del bus.
- Coloca una señal en las líneas de control adecuadas para indicar que desea leer.
- La memoria responde y coloca el valor de la palabra en las líneas de datos para permitir la lectura de esta por parte de la cpu solicitante.

Para *grabar* el procedimiento es similar.

Solo existe una memoria, la cual presenta la propiedad de la ***coherencia****:*

- Las modificaciones hechas por una cpu se reflejan *de inmediato* en las subsiguientes lecturas de la misma o de otra cpu.

El problema de este esquema es que *el bus tiende a sobrecargarse* y el rendimiento a disminuir drásticamente; la solución es añadir una *memoria caché de alta velocidad entre la cpu y el bus:*

- El caché guarda las palabras de acceso reciente.
- Todas las solicitudes de la memoria pasan a través del caché.
- Si la palabra solicitada se encuentra en el caché:
    - El caché responde a la cpu.
    - No se hace solicitud alguna al bus.

[1] Ver Figura 7.1 de la página 218 [25, Tanenbaum].

- Si el caché es lo bastante grande:
  - La *"tasa de encuentros"* será alta y la cantidad de tráfico en el bus por cada cpu disminuirá drásticamente.
  - Permite *incrementar el número de cpu.*

Un importante problema debido al uso de cachés es el de la *"incoherencia de la memoria"*:

- Supongamos que las cpu *"A"* y *"B"* leen la misma palabra de memoria en sus respectivos cachés.
- *"A"* escribe sobre la palabra.
- Cuando *"B"* lee esa palabra, obtiene un valor anterior y no el valor recién actualizado por *"A"*.

Una solución consiste en lo siguiente:

- Diseñar las caché de tal forma que *cuando una palabra sea escrita al caché, también sea escrita a la memoria.*
- A esto se denomina *"caché de escritura".*
- No causa tráfico en el bus el uso de *"caché para la lectura".*
- Sí causa trafico en el bus:
  - El no uso de caché para la lectura.
  - Toda la escritura.

Si *todos los cachés* realizan un monitoreo constante del bus:

- Cada vez que un caché observa una escritura a una dirección de memoria presente en él, puede eliminar ese dato o actualizarlo en el caché con el nuevo valor.
- Estos cachés se denominan *"cachés monitores".*

Un diseño con cachés monitores y de escritura es coherente e invisible para el programador, por lo que es muy utilizado en multiprocesadores basados en buses.

7.7 Multiprocesadores con Conmutador

El esquema de *multiprocesadores con base en buses* resulta apropiado para *hasta aproximadamente 64 procesadores [25, Tanenbaum].*

Para superar esta cifra es necesario un *método distinto de conexión* entre procesadores (cpu) y memoria.

Una posibilidad es dividir la memoria en módulos y conectarlos a las cpu con un *"conmutador de cruceta" (cross-bar switch):*

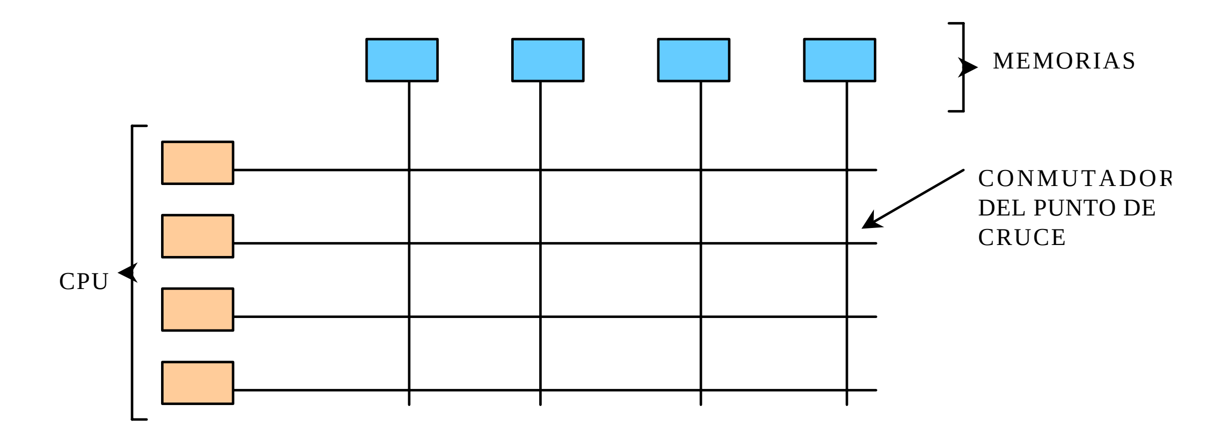


Figura 7.2: Conmutador de cruceta.

- Cada cpu y cada memoria tiene una conexión que sale de él.
- En cada intersección está un *"conmutador del punto de cruce" (crosspoint switch)* electrónico que el *hardware* puede abrir y cerrar:
  - Cuando una cpu desea tener acceso a una memoria particular, el conmutador del punto de cruce que los conecta se cierra momentáneamente.
- La virtud del conmutador de cruceta es que *muchas cpu pueden tener acceso a la memoria al mismo tiempo:*
  - Aunque no a la misma memoria simultáneamente.
- Lo *negativo* de este esquema es el *alto número de conmutadores:*
  - Para *"n"* cpu y *"n"* memorias se necesitan *"n" x "n"* conmutadores.[2]

El número de conmutadores del esquema anterior puede resultar prohibitivo:

- Otros esquemas precisan *menos conmutadores*, por ej., la *"red omega":*[3]
  - ∗ Posee conmutadores 2 x 2:
    - · Cada uno tiene 2 entradas y 2 salidas.
    - · Cada conmutador puede dirigir cualquiera de las entradas en cualquiera de las salidas.
    - · Eligiendo los estados adecuados de los conmutadores, cada cpu podrá tener acceso a cada memoria.
  - ∗ Para *"n" cpu y "n" memorias* se precisan:
    - · *"n"* etapas de conmutación.

[2]Ver Figura 7.2 de la página 220 [25, Tanenbaum].
[3]Ver Figura 7.3 de la página 221 [25, Tanenbaum].

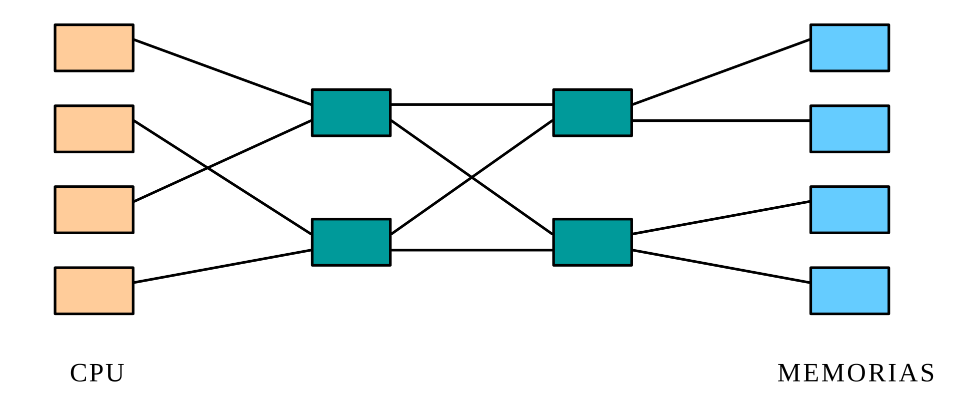


Figura 7.3: Red omega de conmutación.

· Cada etapa tiene *log* $_2$ *n* conmutadores para un total de *n log* $_2$ *n* conmutadores; este número es menor que *"n" x "n"* del esquema anterior, pero *sigue siendo muy grande* para *"n"* grande.[4]

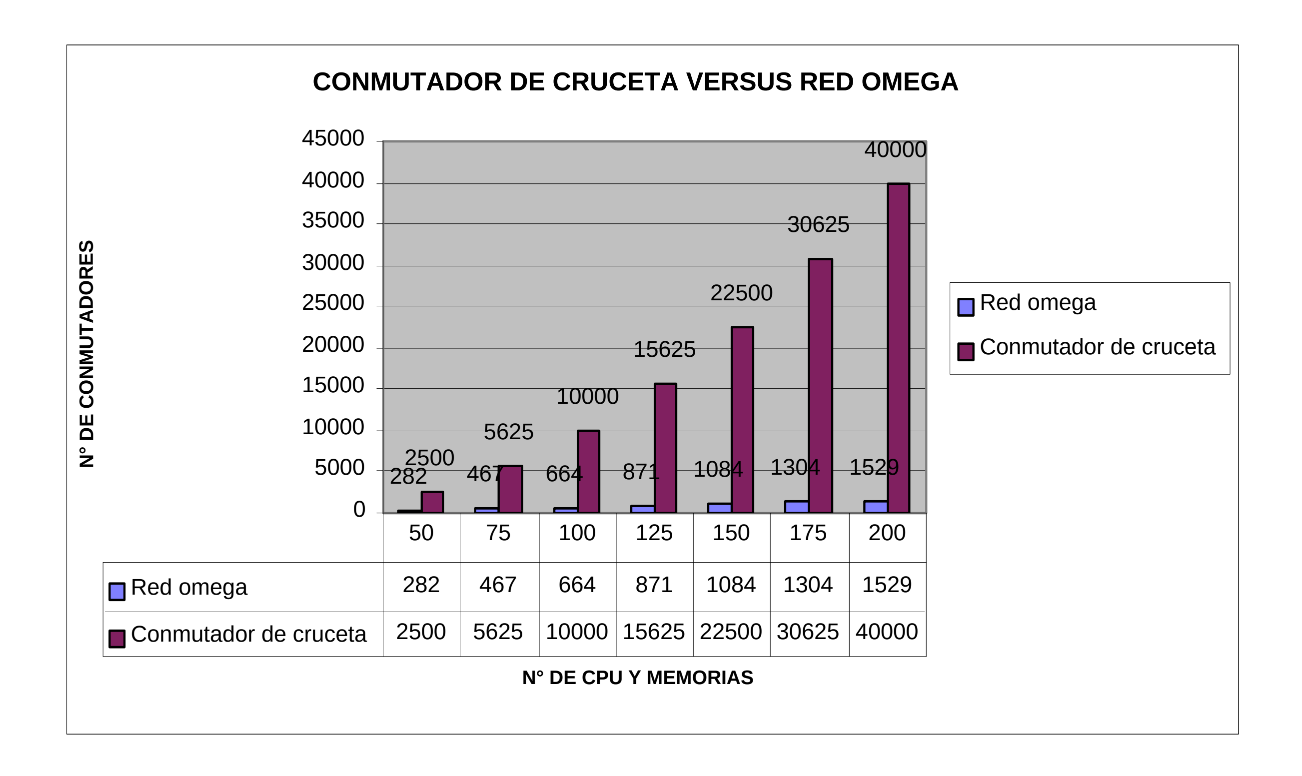
n	**log $_2$ n**	**n * log $_2$ n**	**n * n**
50	5,64385619	282	2.500
75	6,22881869	467	5.625
100	6,64385619	664	10.000
125	6,96578428	871	15.625
150	7,22881869	1.084	22.500
175	7,45121111	1.304	30.625
200	7,64385619	1.529	40.000
1.024	10	10.240	1.048.576

Tabla 7.1: Conmutador de cruceta versus red omega.

Un *problema importante en la red omega* es el retraso:

- Ej.: si *"n"* = 1024 existen según la tabla anterior:
  - 10 etapas de conmutación de la cpu a la memoria.
  - 10 etapas para que la palabra solicitada de la memoria regrese.
  - Si la cpu es de 50 mhz, el tiempo de ejecución de una instrucción es de 20 nseg.
  - Si una solicitud de la memoria debe recorrer 20 etapas de conmutación (10 de ida y 10 de regreso) en 20 nseg:
    - ∗ El tiempo de conmutación debe ser de 1 nseg.
    - ∗ El multiprocesador de 1024 cpu necesitará 10240 conmutadores de 1 nseg.

[4] Ver Tabla 7.1 de la página 221 y Figura 7.4 de la página 222.



	50	75	100	125	150	175	200
Red omega	282	467	664	871	1084	1304	1529
Conmutador de cruceta	2500	5625	10000	15625	22500	30625	40000

Figura 7.4: Conmutador de cruceta versus red omega.

∗ El costo será alto.

Otra *posible solución* son los esquemas según *sistemas jerárquicos:*

- Cada cpu tiene asociada cierta *memoria local.*
- El acceso será muy rápido a la propia memoria local y más lento a la memoria de las demás cpu.
- Esto se denomina esquema o *"máquina NUMA"* (Acceso No Uniforme a la Memoria):
  - Tienen un mejor tiempo promedio de acceso que las máquinas basadas en redes omega.
  - La colocación de los programas y datos en memoria es crítica para *lograr que la mayoría de los accesos sean a la memoria local de cada cpu.*

7.8 Multicomputadoras con Base en Buses

Es un esquema *sin memoria compartida [25, Tanenbaum].*

Cada cpu tiene una *conexión directa con su propia memoria local.*

Un *problema importante* es la forma en que *las cpu se comuniquen entre sí.*

El tráfico es solo entre una cpu y otra; el volumen de tráfico será varios órdenes de magnitud menor que si se utilizara la red de interconexión para el tráfico cpu - memoria.

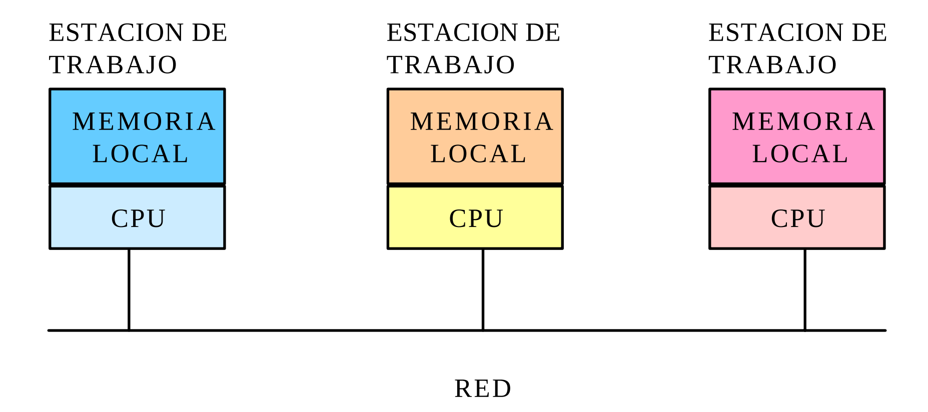


Figura 7.5: Multicomputadora que consta de estaciones de trabajo en una LAN.

Topológicamente es un esquema similar al del multiprocesador basado en un bus.

Consiste generalmente en una colección de estaciones de trabajo en una *LAN (red de área local).*[5]

7.9 Multicomputadoras con Conmutador

Cada cpu tiene *acceso directo y exclusivo a su propia memoria particular [25, Tanenbaum].*

Existen *diversas topologías*, las más comunes son la **retícula** y el **hipercubo.**

Las *principales características de las retículas* son:

- Son fáciles de comprender.
- Se basan en las tarjetas de circuitos impresos.
- Se adecúan a problemas con una naturaleza bidimensional inherente (teoría de gráficas, visión artificial, etc.).[6]

Las *principales características del hipercubo* son:

- Es un cubo *"n"* - dimensional.
- En un *hipercubo de dimensión 4:*
  - Se puede considerar como dos cubos ordinarios, cada uno de ellos con 8 vértices y 12 aristas.
  - Cada vértice es un cubo.
  - Cada arista es una conexión entre 2 cpu.
  - Se conectan los vértices correspondientes de cada uno de los cubos.

[5] Ver Figura 7.5 de la página 223 [25, Tanenbaum].
[6] Ver Figura 7.6 de la página 224 [25, Tanenbaum].

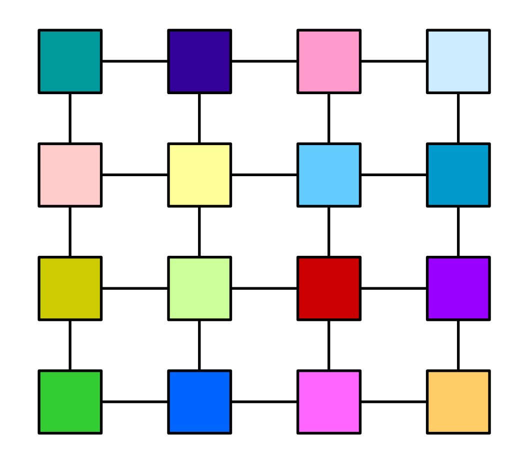


Figura 7.6: Retícula.

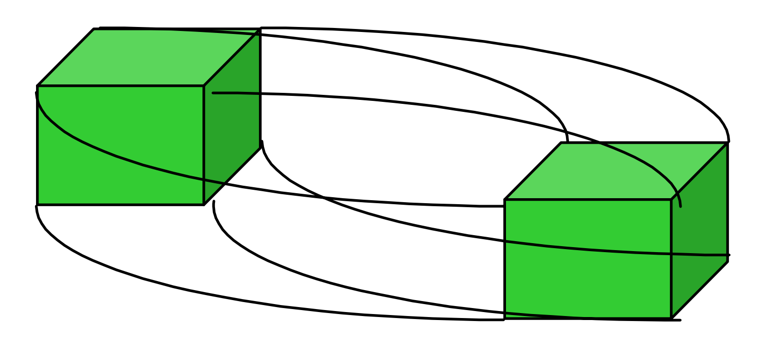


Figura 7.7: Hipercubo de dimensión 4.

- En un *hipercubo de dimensión 5:*
  - Se deberían añadir dos cubos conectados entre sí y conectar las aristas correspondientes en las dos mitades, y así sucesivamente.
- En un *hipercubo de "n" dimensiones:*
  - Cada cpu tiene *"n"* conexiones con otras cpu.
  - La complejidad del cableado aumenta en proporción logarítmica con el tamaño.
  - Solo se conectan los procesadores vecinos más cercanos:
    - Muchos mensajes deben realizar varios saltos antes de llegar a su destino.
    - La trayectoria más grande crece en forma logarítmica con el tamaño:
      - En la retícula crece como la raíz cuadrada del número de cpu.
  - Con la tecnología actual ya se pueden producir hipercubos de 16.384 cpu.[7]

[7] Ver Figura 7.7 de la página 224 [25, Tanenbaum].

7.10 Conceptos de Software

La importancia del software supera frecuentemente a la del hardware [25, Tanenbaum].

La imagen que un sistema presenta queda determinada en gran medida por el software del S. O. y no por el hardware.

Los *S. O.* no se pueden encasillar fácilmente, como el hardware, pero se los puede *clasificar en dos tipos:*

- *Débilmente acoplados.*
- *Fuertemente acoplados.*

El software *débilmente acoplado* de un sistema distribuido:

- Permite que las *máquinas y usuarios sean independientes entre sí* en lo fundamental.
- Facilita que interactúen en cierto grado cuando sea necesario.
- Los equipos individuales se distinguen fácilmente.

Combinando los distintos tipos de *hardware distribuido con software distribuido* se logran distintas soluciones:

- No todas interesan desde el punto de vista funcional del usuario:
  - Ej.: un multiprocesador es un multiprocesador:
    - No importa si utiliza un bus con cachés monitores o una red omega.

7.11 Sistemas Operativos de Redes

Una posibilidad es el *software débilmente acoplado en hardware débilmente acoplado [25, Tanenbaum]:*

- Es una solución muy utilizada.
- Ej.: una red de estaciones de trabajo conectadas mediante una LAN.

Cada usuario tiene una *estación de trabajo para su uso exclusivo:*

- Tiene su propio S. O.
- La mayoría de los requerimientos se resuelven localmente.
- Es posible que un usuario se conecte de manera remota con otra estación de trabajo:
  - Mediante un comando de *"login remoto".*
  - Se convierte la propia *estación de trabajo* del usuario en una *terminal remota* enlazada con la máquina remota.
  - Los comandos se envían a la máquina remota.

- La salida de la máquina remota se exhibe en la pantalla local.

- Para alternar con otra máquina remota, primero hay que desconectarse de la primera:
  - En cualquier instante solo se puede utilizar una máquina.

- Las redes también disponen de un comando de copiado remoto de archivos de una máquina a otra:
  - Requiere que el usuario conozca:
    * La posición de todos los archivos.
    * El sitio donde se ejecutan todos los comandos.

Una *mejor solución* consiste en un *sistema de archivos global compartido*, accesible desde *todas* las estaciones de trabajo:

- Una o varias máquinas soportan al **sistema de archivos:**
  - Son los **"servidores de archivos".**

Los *"servidores de archivos":*

- Aceptan solicitudes de los programas de usuarios:
  - Los *programas* se ejecutan en las máquinas no servidoras, llamadas **"clientes".**
  - Las solicitudes se examinan, se ejecutan y la respuesta se envía de regreso.

- Generalmente tienen un *sistema jerárquico de archivos.*

Las estaciones de trabajo pueden *importar* o *montar* estos sistemas de archivos:

- Se incrementan sus *sistemas de archivos locales.*

- Se pueden montar los servidores en lugares diferentes de sus respectivos sistemas de archivos:
  - Las rutas de acceso a un determinado archivo pueden ser *diferentes* para las distintas estaciones.
  - Los distintos clientes tienen un *punto de vista distinto* del sistema de archivos.
  - El nombre de un archivo depende:
    * Del lugar desde el cual se tiene acceso a él.
    * De la configuración del sistema de archivos.

El S. O. de este tipo de ambiente debe:

- Controlar las estaciones de trabajo en lo individual.

- Controlar a los servidores de archivo.
- Encargarse de la comunicación entre los servidores.

Todas las máquinas *pueden* ejecutar el mismo S. O., pero esto no es necesario.

Si los clientes y los servidores ejecutan diversos S. O., como mínimo deben *coincidir en el formato y significado de todos los mensajes* que podrían intercambiar.

Esquemas como este se denominan **"sistema operativo de red":**

- Cada máquina tiene un alto grado de *autonomía.*
- Existen *pocos requisitos* a lo largo de todo el sistema.

7.11.1 NFS: Network File System

Es uno de los más conocidos y aceptado como **sistema operativo de red [25, Tanenbaum].**

Fue un desarrollo de Sun Microsystems, soportado también por distintos fabricantes:

- Surgió para UNIX pero se amplió a otros S. O. (ej.: MS - DOS).
- Soporta *sistemas heterogéneos*, por ej.: clientes de MS - DOS que hagan uso de servidores UNIX.
- Los equipos pueden ser también de *hardware heterogéneo.*

Los aspectos más interesantes son los relacionados con:

- *La arquitectura.*
- *El protocolo.*
- *La implantación.*

La Arquitectura de NFS

La idea fundamental es permitir que una colección arbitraria de clientes y servidores compartan un sistema de archivos común.

Generalmente todos los clientes y servidores están en la misma LAN, pero esto no es necesario; por ello se puede ejecutar NFS en una *WAN ("red de área amplia").*

NFS permite que *cada máquina* sea *un cliente y un servidor* al mismo tiempo.

Cada servidor de NFS *exporta uno o varios de sus directorios* (y subdirectorios dependientes) para el acceso por parte de clientes remotos.

Los clientes tienen acceso a los directorios exportados mediante el *montaje:*

- Cuando un cliente monta un directorio (remoto), este se convierte en parte de su jerarquía de directorios.

Un cliente sin disco puede montar un archivo remoto en su directorio raíz; esto produce un *sistema de archivos* soportado en su totalidad en un *servidor remoto.*

Las estaciones de trabajo que no poseen discos locales pueden montar directorios remotos en donde lo deseen, en la parte superior de su jerarquía de directorios local; esto produce un *sistema de archivos* que es *en parte local y en parte remoto.*

Si dos o más clientes *montan el mismo directorio* al mismo tiempo:

- Se pueden comunicar al compartir archivos en sus directorios comunes.
- No hay que hacer nada especial para lograr compartir los archivos.

Los *archivos compartidos* figuran en la jerarquía de directorios de varias máquinas y se los puede leer o escribir de la manera usual.

Protocolos de NFS

Uno de los *objetivos de NFS* es:

- Soportar un *sistema heterogéneo* en donde los clientes y servidores podrían ejecutar *distintos S. O. en hardware diverso*, por ello es esencial que la *interfaz* entre los clientes y los servidores esté bien definida.

NFS logra este objetivo definiendo dos **"protocolos cliente - servidor":**

- Un **"protocolo"** es un conjunto de:
  - *Solicitudes* que envían los clientes a los servidores.
  - *Respuestas* que envían los servidores de regreso a los clientes.

Un **"protocolo de NFS"** maneja el *montaje*.
Un *cliente* puede:

- Enviar el nombre de una ruta de acceso a un servidor.
- Solicitar el permiso para montar ese directorio en alguna parte de su jerarquía de directorios.

Si el nombre de la ruta de acceso es *válido* y el directorio especificado ha sido *exportado:*

- El servidor regresa un *"asa de archivo" (file handle)* al cliente:
  - Contiene campos que identifican:
    * De manera única el tipo de sistema de archivos, el disco, el número de nodo-i del directorio.
    * La información relativa a la seguridad.
  - Es utilizada en llamadas posteriores para la lectura o escritura de archivos en el directorio montado.

Algunos S. O. soportan la alternativa del *"automontaje":*

- Permite que un conjunto de directorios remotos quede *asociado* con un directorio local.
- Ninguno de los directorios remotos se monta durante el arranque del cliente.
- La primera vez que se abra un archivo remoto, el S. O. envía un mensaje a los servidores:
  - Los servidores responden y se monta su directorio.
- Las principales *ventajas sobre el montaje estático* son:
  - Se evita el trabajo de contactar servidores y montar directorios que *no son requeridos de inmediato.*
  - Si el cliente puede utilizar varios servidores en paralelo, se puede tener:
    * Cierta tolerancia a fallas.
    * Mejorar el rendimiento.

NFS no da soporte a la *duplicación* de archivos o directorios.
Otro **"protocolo de NFS"** es para el *acceso a los directorios y archivos.*
Los *clientes* pueden:

- Enviar mensajes a los servidores para el manejo de los directorios y la lectura o escritura de archivos.
- Tener acceso a los atributos de archivo, tales como su modo, tamaño y fecha de la última modificación.

NFS soporta **"servidores sin estado":**

- *No mantienen la información de estado relativa a los archivos abiertos.*
- Si un servidor falla y arranca rápidamente, no se pierde información acerca de los archivos abiertos y los programas cliente no fallan.

El *"sistema de archivos remotos" (RFS)* del Sistema V de UNIX no funciona así, sino que:

- El servidor lleva un registro del hecho que cierto archivo está abierto y la posición actual del lector.
- Si un servidor falla y vuelve a arrancar rápidamente:
  - Se pierden todas las conexiones abiertas.
  - Los programas cliente fallan.

En un *"servidor sin estado"*, como NFS:

- Los bloqueos no tienen que asociarse con los archivos abiertos y el servidor no sabe cuáles archivos están abiertos.

- Se necesita un *mecanismo adicional independiente* para controlar el *bloqueo.*

NFS utiliza el *esquema de protección de UNIX*, con los bits *"rwx"* para el propietario, grupo y otros.

Se puede utilizar la criptografía de claves públicas para dar validez al cliente y el servidor en cada solicitud y respuesta; el cliente malicioso no puede personificar a otro cliente, ya que no conoce su clave secreta.

Las claves utilizadas para la autentificación, así como otra información, están contenidas en el **NIS**:

- *"Network Information Service"*: Servicio de Información de la Red.

- Almacena parejas (clave, valor).

- Cuando se proporciona una clave, regresa el valor correspondiente.

- Almacena la asociación de:

  - Los nombres de los usuarios con las contraseñas (cifradas).
  - Los nombres de las máquinas con las direcciones en la red y otros elementos.

Implantación de NFS

La *implantación del código* del cliente y el servidor es *independiente de los protocolos NFS.*

Una implementación que suele tomarse como referencia es la de Sun, que consta de tres capas.[8]

La *capa superior* es la de *llamadas al sistema:*

- Maneja las llamadas del tipo *open, read* y *close.*

- Analiza la llamada y verifica los parámetros.

- Llama a la *segunda capa*: capa del *sistema virtual de archivos: virtual file system: VFS.*

La *capa VFS* mantiene una *tabla con una entrada por cada archivo abierto* que es análoga a la tabla de *nodos-i* para los archivos abiertos en UNIX.

La capa VFS tiene una entrada por cada archivo abierto:

- Se la llama *nodo-v (nodo-i virtual).*

- Los nodos-v se utilizan para indicar si el archivo es *local o remoto.*

- Para los archivos remotos, poseen la información suficiente como para tener acceso a ellos.

[8] Ver Figura 7.8 de la página 233 [25, Tanenbaum].

Para *montar un sistema remoto de archivos*, el administrador del sistema llama al programa *mount :*

- Utiliza la información del directorio remoto, el directorio local donde será montado y otros datos adicionales.
- Con el nombre del directorio remoto por montar se descubre el nombre de la máquina donde se localiza dicho directorio.
- Se verifica si el directorio existe y si está disponible para su montaje remoto.

El núcleo:

- Construye un *nodo-v* para el directorio remoto.
- Pide el código del cliente NFS para crear un *nodo-r* (nodo-i remoto) en sus tablas internas.

El nodo-v apunta al nodo-r.

Cada *nodo-v* de la capa VFS contendrá en última instancia un *apuntador a un nodo-i* en el S. O. local.

Es posible ver desde el *nodo-v* si un archivo o directorio es local o remoto y, si es remoto, encontrar su asa de archivo.

Todo archivo o directorio abierto tiene un *nodo-v* que apunta a un *nodo-r* o a un *nodo-i.*

Por razones de eficiencia las transferencias entre cliente y servidor se hacen en bloques grandes, generalmente de 8k:

- Luego de haber recibido la capa VFS del cliente el bloque necesario, emite la solicitud del siguiente bloque; esto se denomina *lectura adelantada (read ahead).*

Un criterio similar se sigue con la *escritura:*

- Antes de ser enviados al servidor los datos se acumulan en forma local:
  - Hasta completar cierta cantidad de bytes, o
  - Hasta que se cierra el archivo.

Otra técnica utilizada para *mejorar el rendimiento* es el *ocultamiento* o *caching:*

- Los *servidores* ocultan los datos para evitar el acceso al disco.
- Esto es invisible para los clientes.

Los *clientes* mantienen dos cachés:

- Uno para los atributos de archivo (nodos-i).
- Otro para los datos del archivo.

Cuando se necesita un nodo-i o un bloque del archivo:

- Primero se verifica si la solicitud se puede satisfacer mediante el caché del cliente, con esto se evita el tráfico en la red.

Un *problema importante del caching* es que el *caché no es coherente*; ej.:

- Dos clientes ocultan el mismo bloque del archivo.
- Uno de ellos lo modifica.
- Cuando el otro lee el bloque, obtiene el valor antiguo.
- Para mitigar este problema, la implantación de NFS:
  - Asocia a cada bloque caché un temporizador (timer).
  - Cuando el timer expira, la entrada se descarta.
  - Generalmente los tiempos son de:
    * 3 segundos para bloques de datos.
    * 30 segundos para bloques de directorio.
- Al abrir un archivo con caché se envía un mensaje al servidor para revisar la hora de la última modificación.
- Se determina si la copia del caché es válida o debe descartarse, utilizando una nueva copia del servidor.
- El temporizador del caché expira cada 30 segundos y todos los bloques modificados en el caché se envían al servidor.

Resumiendo:

- *NFS solo trata el sistema de archivos.*
- *NFS no hace referencia a otros aspectos, como la ejecución de un proceso.*
- *NFS se ha difundido ampliamente, a pesar de todo.*

7.12 Sistemas Realmente Distribuidos

NFS es un ejemplo de software débilmente acoplado en hardware débilmente acoplado [25, Tanenbaum]:

- Cada computadora puede ejecutar su propio S. O.
- Solo se dispone de un *sistema compartido de archivos.*
- El tráfico cliente - servidor debe obedecer los protocolos NFS.

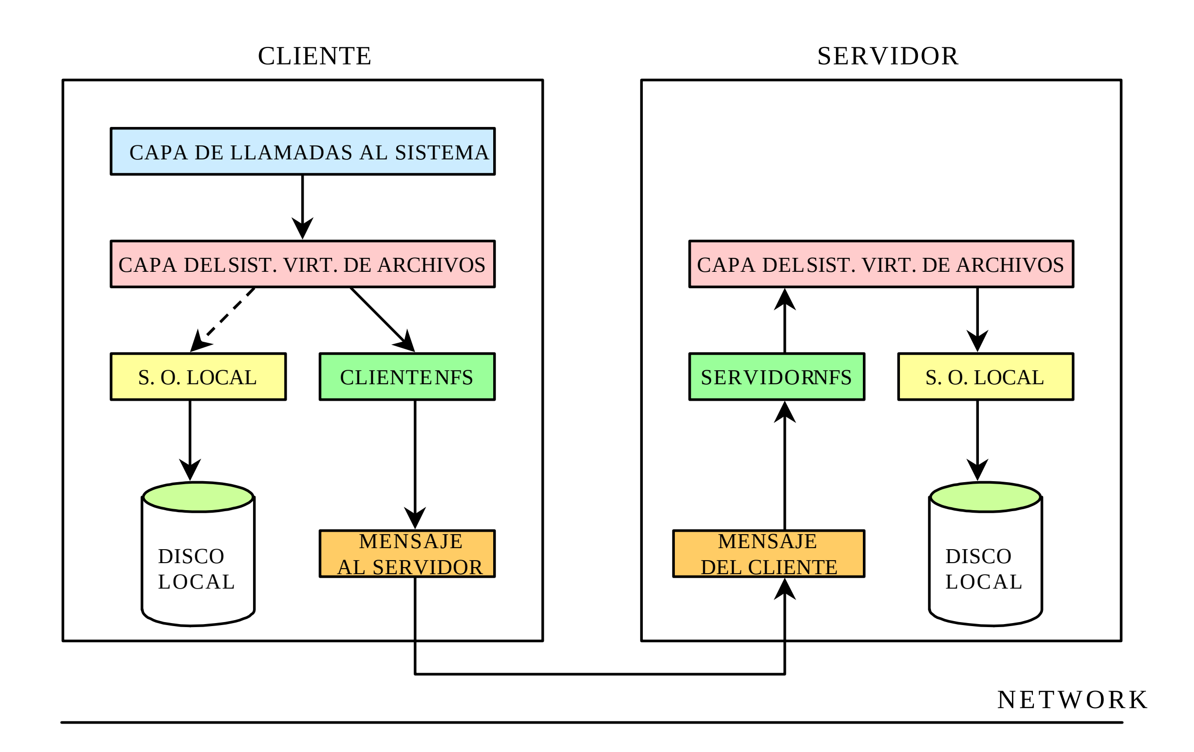


Figura 7.8: Estructura de capas de NFS.

Las multicomputadoras son un ejemplo de software fuertemente acoplado en hardware débilmente acoplado:

- Crean la *ilusión* de que toda la red de computadoras es *un solo sistema de tiempo compartido*, en vez de una colección de máquinas diversas.

Un sistema distribuido es aquel que se ejecuta en una colección de máquinas sin memoria compartida, pero que aparece ante sus usuarios como una sola computadora:

- A esta propiedad se la conoce como la **imagen de un único sistema.**

También se define un sistema distribuido como aquel que se ejecuta en una colección de máquinas enlazadas mediante una red pero que actúan como un ***uniprocesador virtual****.*

Algunas de las *características* de los sistemas distribuidos son las siguientes:

- Debe existir un *mecanismo de comunicación global* entre los procesos:
  - Cualquier proceso debe poder comunicarse (intercambiar información) con cualquier otro.
- No tiene que haber:
  - Distintos mecanismos en distintas máquinas.
  - Distintos mecanismos para la comunicación local o la comunicación remota.

- Debe existir un *esquema global de protección.*
- La *administración de procesos debe ser la misma* en todas parte.
- Se debe tener una *misma interfaz de llamadas al sistema* en todas partes:
  - Es normal que se ejecuten núcleos idénticos en todas las cpu del sistema.
- Es necesario un *sistema global de archivos.*

7.13 Sistemas de Multiprocesador con Tiempo Compartido

Corresponde a software fuertemente acoplado en hardware fuertemente acoplado [25, Tanenbaum].

Los ejemplos más comunes de propósito general son los *multiprocesadores:*

- Operan *como un sistema de tiempo compartido*, pero con *varias cpu* en vez de una sola.
- Externamente un multiprocesador con 32 cpu de 3 mips actúa de manera muy parecida a una sola cpu de 96 mips; 1 mips: 1.000.000 de instrucciones por segundo.
- Se corresponde con la *imagen de un único sistema.*

La característica clave es la existencia de una sola cola para ejecución:[9]

- Una lista de todos los procesos en el sistema que no están bloqueados en forma lógica y listos para su ejecución.
- La *cola de ejecución* es una estructura de datos contenida en la *memoria compartida.*

Los programas de los procesos están en la memoria compartida, también el S. O.

El *planificador (de procesos)* del S. O. se ejecuta como una *"región crítica"*, con ello se evita que dos cpu elijan el mismo proceso para su ejecución inmediata.

Cuando un *proceso se asigna a un procesador:*

- Encuentra que el caché del procesador está ocupado por palabras de memoria que pertenecen a aquella parte de la memoria compartida que contiene al programa del proceso anterior.
- Luego de un breve lapso se habrán reemplazado por el código y los datos del programa del proceso asignado a ese procesador.

Ninguna cpu tiene memoria local, es decir que todos los programas se almacenan en la *memoria global compartida.*

Si todas las cpu están inactivas en espera de e / s y un proceso está listo para su ejecución:

[9] Ver Figura 7.9 de la página 235 [25, Tanenbaum].

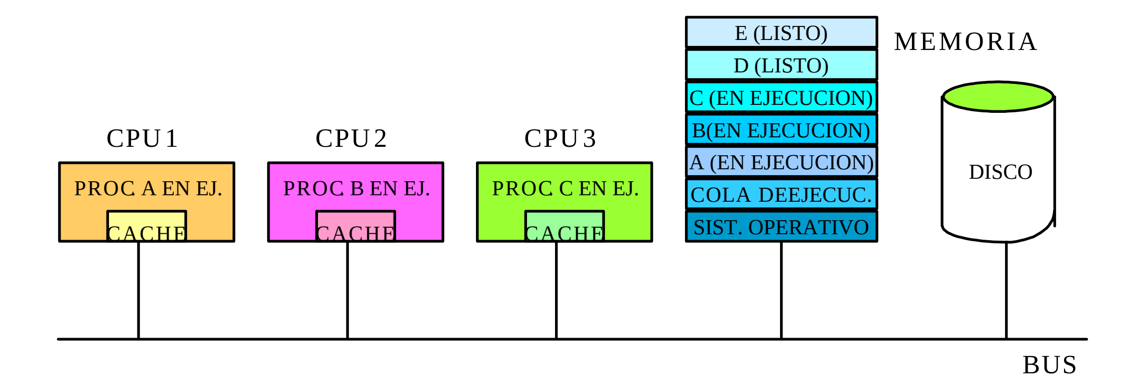


Figura 7.9: Un multiprocesador con una sola cola de ejecución.

- Es conveniente asignarlo a la cpu que se utilizó por última vez (para ese proceso):
  - La hipótesis es que ningún otro proceso utilizó esa cpu desde entonces *(hipótesis de Vaswani y Zahorjan).*

Si un proceso se bloquea en espera de e / s en un multiprocesador, el S. O. puede:

- Suspenderlo.
- Dejarlo en *"espera ocupada":*
  - Es aplicable cuando la mayoría de la e / s se realiza en *menos tiempo* del que tarda un cambio entre los procesos.
  - El proceso conserva su procesador por algunos milisegundos en espera de que la e / s finalice:
    * Si se agota el tiempo de espera y no ha finalizado la e / s, se realiza una conmutación de procesos.

Generalmente se dispondrá de un *sistema de archivos tradicional, con un único caché:*

- Globalmente considerado es similar al sistema de archivos de un único procesador.

7.14 Aspectos del Diseño

La comparación de las tres *principales formas* de organizar *"n"* cpu se puede resumir en la Tabla 7.2 de la página 236 [25, Tanenbaum].

Los *aspectos claves en el diseño de S. O. distribuidos* son:

- *Transparencia.*
- *Flexibilidad.*

Elemento	**S. O. de red**	**S. O. distribuido**	**S. O. de multiprocesador**
¿Se ve como un uniprocesador virtual?	No	Sí	Sí
¿Todas tienen que ejecutar el mismo S. O.?	No	Sí	Sí
¿Cuántas copias del S. O. existen?	n	n	1
¿Cómo se logra la comunicación?	Archivos compartidos	Mensajes	Memoria compartida
¿Se requiere un acuerdo en los protocolos de la red?	Sí	Sí	No
¿Existe una única cola de ejecución?	No	No	Sí
¿Existe una semántica bien definida para los archivos compartidos?	Por lo general No	Sí	Sí

Tabla 7.2: Comparación de tres formas distintas de organizar *"n"* cpu.

- *Confiabilidad.*
- *Desempeño.*
- *Escalabilidad.*

7.15 Transparencia

Un aspecto muy importante es la *forma* de lograr la *imagen de un único sistema [25, Tanenbaum].*

Los usuarios *deben percibir* que la colección de máquinas conectadas son *un sistema de tiempo compartido de un solo procesador:*

- Un sistema que logre este objetivo se dice que es *transparente.*

Desde el *punto de vista de los usuarios*, la transparencia se logra cuando:

- Sus pedidos se satisfacen con ejecuciones en paralelo en distintas máquinas.
- Se utilizan una variedad de servidores de archivos.
- El usuario no necesita saberlo ni notarlo.

La transparencia desde el *punto de vista de los programas* significa diseñar la interfaz de llamadas al sistema de modo que no sea visible la existencia de varios procesadores.

No es transparente un sistema donde el *acceso a los archivos remotos* se realice mediante:

- El establecimiento explícito de una conexión en la red con un servidor remoto.
- El envío posterior de mensajes, donde el acceso a los servicios remotos será distinto al acceso a los servicios locales.

Existen distintos *tipos de transparencia* en un sistema distribuido:

- *De localización:* los usuarios no pueden indicar la localización de los recursos.
- *De migración:* los recursos se pueden mover a voluntad sin cambiar sus nombres.
- *De réplica:* los usuarios no pueden indicar el número de copias existentes.
- *De concurrencia:* varios usuarios pueden compartir recursos de manera automática.
- *De paralelismo:* las actividades pueden ocurrir en paralelo sin el conocimiento de los usuarios.

7.16 Flexibilidad

La flexibilidad es de *fundamental importancia [25, Tanenbaum].*

Existen dos *escuelas de pensamiento* en cuanto a la *estructura de los sistemas distribuidos:*[10]

- *Núcleo monolítico:*
  - Cada máquina debe ejecutar un núcleo tradicional que proporcione la mayoría de los servicios.
- *Micronúcleo (microkernel):*
  - El núcleo debe proporcionar lo menos posible.
  - El grueso de los servicios del S. O. se debe obtener a partir de los servidores al nivel usuario.

El *núcleo monolítico* es el S. O. centralizado aumentado con:

- Capacidades de red.
- Integración de servicios remotos.

Con *núcleo monolítico:*

- La mayoría de las llamadas al sistema se realizan mediante señalamiento al núcleo:
  - El núcleo realiza el trabajo.
  - El núcleo regresa el resultado al proceso del usuario.

[10] Ver Figura 7.10 de la página 239 [25, Tanenbaum].

- La mayoría de las máquinas tiene discos y administra sus propios sistemas locales de archivos.

El *micronúcleo es más flexible* y proporciona solo cuatro servicios mínimos:

- Un mecanismo de comunicación entre procesos.
- Cierta administración de la memoria.
- Una cantidad limitada de planificación y administración de procesos de bajo nivel.
- Entrada / salida de bajo nivel.

Contrariamente al núcleo monolítico, el *micronúcleo no proporciona* el sistema de archivos, el sistema de directorios, toda la administración de procesos o gran parte del manejo de las llamadas al sistema.

El objetivo es mantener el micronúcleo pequeño.

Todos los demás servicios del S. O. se implementan generalmente como servidores a nivel usuario:

- Para obtener un servicio:
  - El usuario envía un mensaje al servidor apropiado.
  - El servidor realiza el trabajo y regresa el resultado.

Una *importante ventaja* de este método es su *alta modularidad:*

- Existe una interfaz bien definida con cada servicio (conjunto de mensajes que comprende el servidor).
- Cada servicio es igual de accesible para todos los clientes, independientemente de la posición.
- Es fácil implantar, instalar y depurar nuevos servicios, sin necesidad de detener el sistema totalmente.

7.17 Confiabilidad

Un importante *objetivo de los sistemas distribuidos* es que *si una máquina falla, alguna otra debe encargarse del trabajo [25, Tanenbaum].*

La *confiabilidad global teórica* del sistema podría ser el *"or"* booleano de la *confiabilidad de los componentes*; ejemplo:

- Se dispone de 5 servidores de archivos, cada uno con una probabilidad de 0,95 de funcionar en un instante dado.
- La probabilidad de falla simultánea de los 5 es $(0{,}05)^5 = 0{,}000006$.

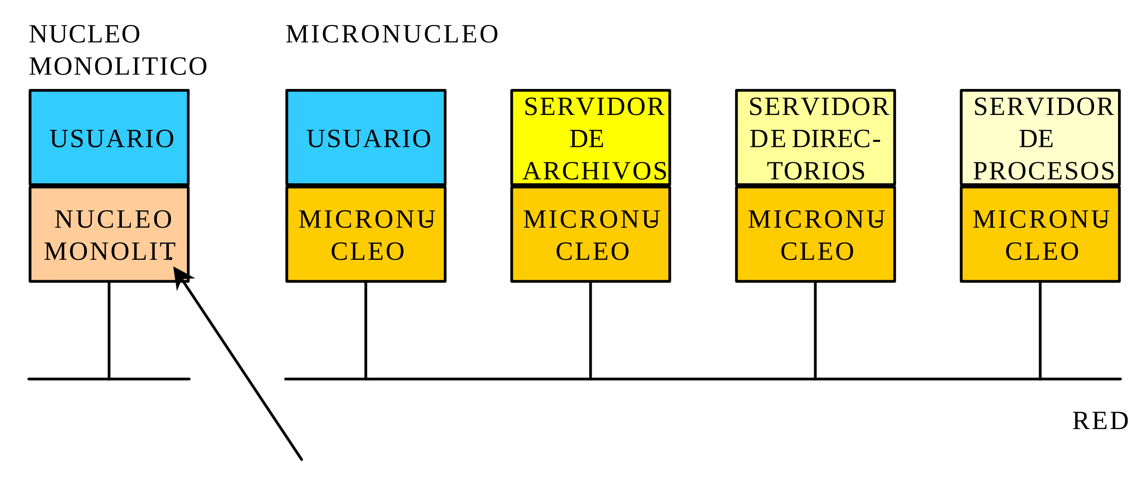


Figura 7.10: Esquema de núcleo monolítico y de micronúcleo.

- La probabilidad de que al menos uno esté disponible es 0,999994.

La *confiabilidad práctica* se ve disminuida ya que muchas veces se requiere que ciertos servidores estén en servicio simultáneamente para que el todo funcione, debido a ello algunos sistemas tienen una disponibilidad más relacionada con el *“and”* booleano de las componentes que con el *“or”* booleano.

Un aspecto de la confiabilidad es la *disponibilidad,* que se refiere a la *fracción de tiempo en que se puede utilizar el sistema.*

La disponibilidad se mejora mediante:

- Un diseño que no exija el funcionamiento simultáneo de un número sustancial de componentes críticos.

- La *redundancia*, es decir la duplicidad de componentes clave del hardware y del software.

Los datos no deben perderse o mezclarse y si los archivos se almacenan de manera redundante en varios servidores, *todas las copias deben ser consistentes.*

Otro aspecto de la confiabilidad general es la *seguridad*, lo que significa que los archivos y otros recursos deben ser protegidos contra el uso no autorizado.

Un aspecto también relacionado con la confiabilidad es la *tolerancia a fallas*, según la cual las fallas se deben ocultar brindando una *recuperación transparente para el usuario*, aunque haya cierta degradación de la performance.

7.18 Desempeño

Cuando se ejecuta una aplicación en un sistema distribuido *no debe parecer peor que su ejecución en un único procesador*, pero esto es difícil de lograr [25, Tanenbaum].

Algunas *métricas del desempeño* son:

- Tiempo de respuesta.
- Rendimiento (número de trabajos por hora).
- Uso del sistema y cantidad consumida de la capacidad de la red.

El problema se complica por el hecho de que la comunicación entre equipos es lenta comparada con:

- La velocidad de proceso.
- La velocidad de la comunicación dentro de un mismo procesador.

Se requiere el uso de protocolos de comunicaciones en los extremos (procesadores) que intervienen en la comunicación, con lo que se incrementa el consumo de ciclos de procesador.

Para *optimizar el desempeño* frecuentemente hay que:

- Minimizar el número de mensajes:
  - La dificultad es que la mejor forma de mejorar el desempeño es tener muchas actividades en ejecución paralela en distintos procesadores, pero esto requiere el envío de muchos mensajes.
- Centralizar el trabajo en una sola máquina:
  - Resulta poco apropiado para un sistema distribuido.

También se debe prestar atención al *tamaño de grano* de todos los cálculos:

- *Paralelismo de grano fino:*
  - Corresponde a trabajos con un gran número de pequeños cálculos y mucha interacción con otros trabajos, debido a ello requieren mucha comunicación que puede afectar el desempeño.
- *Paralelismo de grano grueso:*
  - Corresponde a trabajos con grandes cálculos, poca interacción y pocos datos, por lo tanto requieren poca comunicación y no afectan la performance.

7.19 Escalabilidad

La tendencia indica que el tamaño de los sistemas distribuidos es hacia *cientos de miles y aun decenas de millones de usuarios conectados [25, Tanenbaum].*

Existen *cuellos de botella potenciales* que se debe intentar evitar en los *sistemas distribuidos de gran escala:*

- *Componentes centralizados:*
  - Ej.: un solo servidor de correo para todos los usuarios.
- *Tablas centralizadas:*
  - Ej.: un único directorio telefónico en línea.
- *Algoritmos centralizados:*
  - Ej.: realización de un ruteo con base en la información completa.

Se deben *utilizar algoritmos descentralizados* con las siguientes características:

- Ninguna máquina tiene la información completa acerca del estado del sistema.
- Las máquinas toman decisiones solo en base a la información disponible de manera local.
- El fallo de una máquina no arruina el algoritmo.
- No existe una hipótesis implícita de la existencia de un reloj global.

Capítulo 8

Comunicación en los Sistemas Distribuidos

8.1 Introducción a la Comunicación en los Sistemas Distribuidos

La diferencia más importante entre un sistema distribuido y un sistema de un único procesador es la comunicación entre procesos [25, Tanenbaum].

En un sistema de *un solo procesador* la comunicación supone implícitamente la existencia de la *memoria compartida:*

- Ej.: problema de los productores y los consumidores, donde un proceso escribe en un buffer compartido y otro proceso lee de él.

En un *sistema distribuido no existe la memoria compartida* y por ello toda la naturaleza de la comunicación entre procesos debe replantearse.

Los procesos, para comunicarse, deben apegarse a reglas conocidas como *protocolos.*

Para los *sistemas distribuidos en un área amplia*, estos protocolos toman frecuentemente la forma de *varias capas* y cada capa tiene sus propias metas y reglas.

Los mensajes se intercambian de diversas formas, existiendo muchas opciones de diseño al respecto; una importante opción es la *"llamada a un procedimiento remoto".*

También es importante considerar las posibilidades de *comunicación entre grupos de procesos*, no solo entre dos procesos.

8.2 Protocolos con Capas

Debido a la ausencia de memoria compartida, toda la comunicación en los sistemas distribuidos se basa en la *transferencia de mensajes [25, Tanenbaum].*

Cuando el proceso *"A"* quiere comunicarse con el proceso *"B":*

- Construye un *mensaje* en su propio espacio de direcciones.

- Ejecuta una *llamada al sistema* para que el S. O. busque el mensaje y lo envíe a través de la red hacia *"B".*

- Para evitar el caos, *"A"* y *"B"* deben coincidir en el *significado* de los bits que se envíen.

Los *puntos de acuerdo necesarios* incluyen lo siguiente:

- ¿Cuántos voltios hay que utilizar para un bit "0" y cuántos para un bit "1"?.
- ¿Cómo sabe el receptor cuál es el último bit del mensaje?.
- ¿Cómo puede detectar si un mensaje ha sido dañado o perdido, y qué debe hacer si lo descubre?.
- ¿Qué longitud tienen los números, cadenas y otros elementos de datos y cuál es la forma en que están representados?.

La **ISO** (Organización Internacional de Estándares) desarrolló un *modelo de referencia* que:[1]

- Identifica en forma clara los distintos niveles.
- Estandariza los nombres de los niveles.
- Señala cuál nivel debe realizar cuál trabajo.

Este modelo se denomina *"modelo de referencia para interconexión de sistemas abiertos"* (**ISO OSI** o **modelo OSI**) [26, Tanenbaum].
El *"modelo OSI"* está diseñado para permitir la *comunicación de los sistemas abiertos:*

- Son aquellos preparados para comunicarse con cualquier otro sistema abierto mediante *reglas estándar:*
  - Establecen el formato, contenido y significado de los mensajes recibidos y enviados.
  - Constituyen los **protocolos,** que son *acuerdos en la* ***forma*** *en que debe desarrollarse la comunicación.*[2]

El *"modelo OSI"* distingue entre *dos tipos generales de protocolos:*

- *Orientados hacia las conexiones:*
  - Antes de intercambiar los datos, el emisor y el receptor:
    - Establecen en forma explícita una conexión.
    - Probablemente negocien el protocolo a utilizar.
    - Al finalizar, deben terminar la conexión.
    - El teléfono es un sistema de comunicación orientado hacia la conexión.

[1] Ver Figura 8.1 de la página 246 [25, Tanenbaum].
[2] Ver Figura 8.2 de la página 246 [25, Tanenbaum].

- *Sin conexión:*
  - No es necesaria una configuración de antemano.
  - El emisor transmite el primer mensaje cuando está listo.
  - El depósito de una carta en un buzón es una comunicación sin conexión.

Cada capa proporciona una *interfaz* con la otra capa por encima de ella; la interfaz consiste de un *conjunto de operaciones* para definir el *servicio que la capa está preparada para ofrecer a sus usuarios.*

El *protocolo de la capa "n"* utiliza la información de la capa *"n".*

Cada protocolo de capa se puede cambiar independientemente de los demás:

- Esto es de fundamental importancia.
- Confiere gran flexibilidad.

La colección de protocolos utilizados en un sistema particular se llama una **"suite de protocolo"** o **"pila de protocolo".**

8.3 Introducción al Modelo Cliente - Servidor (C - S)

El *"modelo de la OSI"* es una solución elegante y realmente aplicable en muchos casos, pero tiene un problema [25, Tanenbaum]:

- La existencia de los encabezados genera un *"costo"* adicional de transmisión.
- Cada envío de un mensaje genera:
  - Proceso en media docena de capas.
  - Preparación y agregado de encabezados en el camino hacia "abajo".
  - Eliminación y examen de encabezados en el camino hacia "arriba".

Con enlaces del orden de decenas (o centenas) de miles de bits / segundo y cpu poderosas:

- La carga de procesamiento de los protocolos no es significativa.
- El factor limitante es la capacidad de las líneas.
- Ej.: redes de área extendida (WAN).

Con enlaces del orden de millones de bits / segundo y computadoras personales:

- La carga de procesamiento de los protocolos sí es frecuentemente significativa.
- El factor limitante no es la capacidad de las líneas.
- Ej.: redes de área local (LAN).

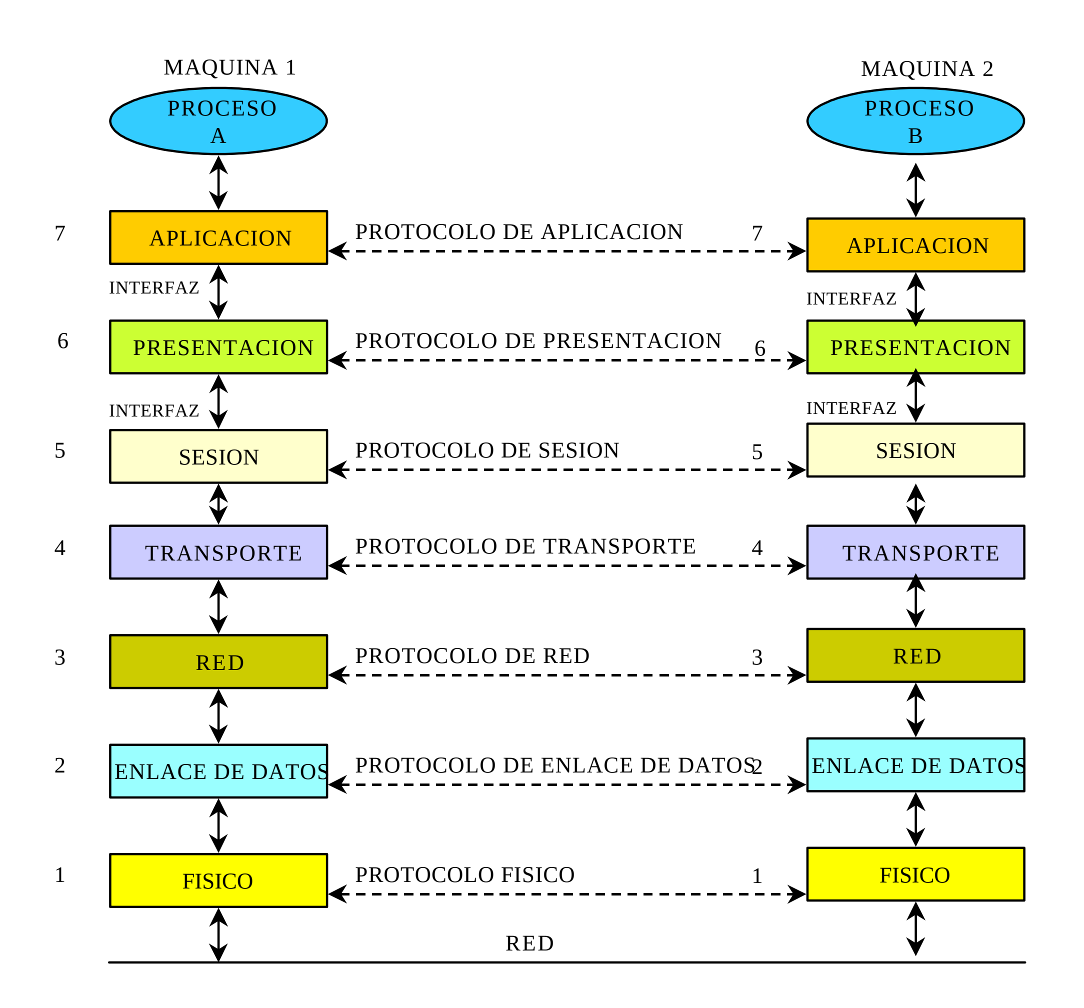


Figura 8.1: Capas, interfaces y protocolos en el modelo OSI.

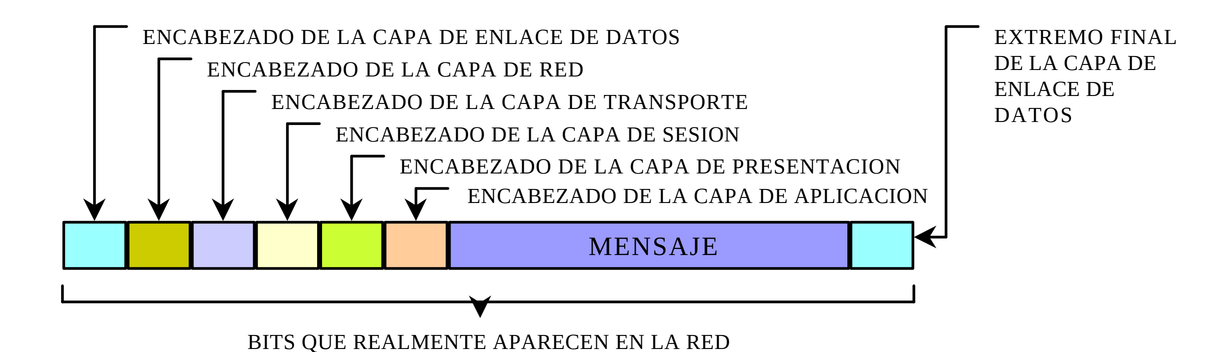


Figura 8.2: Un mensaje típico tal como aparece en la red.

La mayoría de los sistemas distribuidos basados en LAN no utilizan los protocolos de capas completos, sí utilizan un subconjunto de toda una pila de protocolos.

El *"modelo OSI"* no dice nada acerca de la *forma* de estructurar al sistema distribuido.

El *"modelo cliente - servidor"* tiene como idea fundamental la estructuración del S. O. como:

- Un grupo de *procesos en cooperación*, llamados *servidores*, que ofrecen servicios a los usuarios.
- Un grupo de *procesos usuarios* llamados *clientes.*

El **"modelo cliente - servidor"** se basa en un *"protocolo solicitud / respuesta":*

- Es sencillo y sin conexión.
- No es complejo y orientado a la conexión como OSI o TCP / IP.
- El cliente envía un mensaje de solicitud al servidor pidiendo cierto servicio.
- El servidor:
  - Ejecuta el requerimiento.
  - Regresa los datos solicitados o un código de error si no pudo ejecutarlo correctamente.
- No se tiene que establecer una conexión sino hasta que ésta se utilice.
- La pila del protocolo es más corta y por lo tanto más eficiente.
- Si todas las máquinas fuesen idénticas solo se necesitarían tres niveles de protocolos.[3]

8.4 Direccionamiento en C - S

Para que un *cliente* pueda enviar un mensaje a un *servidor*, debe conocer la *dirección* de éste [25, Tanenbaum].

Un *esquema de direccionamiento* se basa en la *dirección de la máquina destinataria del mensaje:*

- Es limitativo si en la máquina destinataria se ejecutan varios procesos, pues no se sabría para cuál de ellos es el mensaje.

Otro *esquema de direccionamiento* se basa en *identificar los procesos destinatarios* en vez de a las máquinas:

- Elimina la ambigüedad acerca de quién es el receptor.
- Presenta el problema de *cómo identificar los procesos:*

[3] Ver Figura 8.3 de la página 248 [25, Tanenbaum].

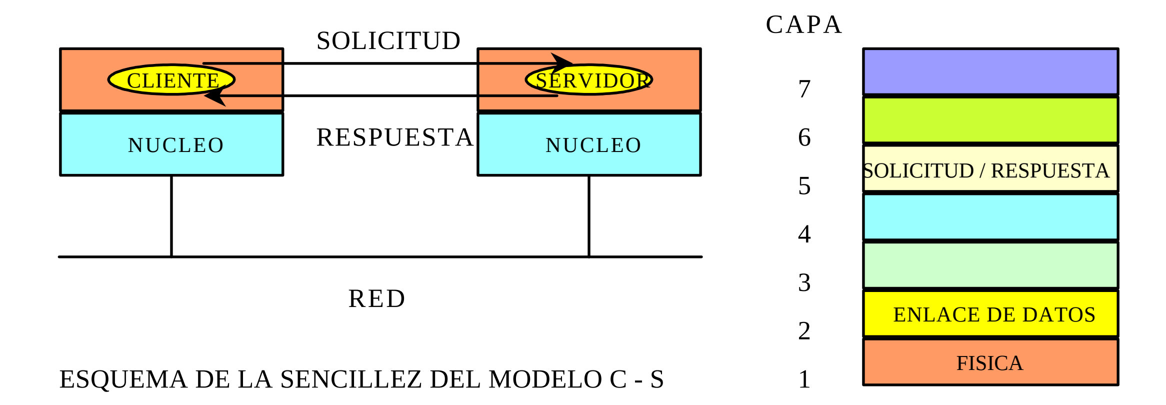


Figura 8.3: Modelo cliente - servidor.

- Una solución es una nomenclatura que incluya la *identificación de la máquina y del proceso:*
  - ∗ No se necesitan coordenadas globales.
  - ∗ Pueden repetirse los nombres de los procesos en distintas máquinas.

Una variante utiliza *machine.local-id* en vez de *machine.process:*

- *local-id* generalmente es un entero aleatorio de 16 o 32 bits.
- Un proceso servidor se inicia mediante una llamada al sistema para indicarle al núcleo que desea escuchar a *local-id.*
- Cuando se envía un mensaje dirigido a *machine.local-id* el núcleo sabe a cuál proceso debe dar el mensaje.

El direccionamiento *machine.process* presenta el serio *inconveniente de que no es transparente:*

- La *transparencia* es uno de los principales objetivos de la construcción de sistemas distribuidos.
- El usuario debe conocer la posición del servidor.
- Un cambio de servidor obliga a cambiar los programas.

Otro *método de direccionamiento* consiste en asignarle a cada proceso una única dirección que no contenga un número de máquina:

- Una forma es mediante un *asignador centralizado de direcciones a los procesos* que mantenga un contador:
  - Al recibir una solicitud de dirección regresa el valor actual del contador y lo incrementa en uno.
- La desventaja es el elemento centralizado.

También existe el *método de dejar que cada proceso elija su propio identificador:*

- En un espacio de direcciones grande y disperso, por ej.: enteros binarios de 64 bits.
- La probabilidad de que dos procesos elijan el mismo número es muy pequeña.
- Existe el problema, para el núcleo emisor, de saber a qué máquina enviar el mensaje:
  - En una LAN, el emisor puede transmitir un *paquete especial de localización* con la dirección del proceso destino.
  - Este paquete de transmisión será recibido por todas las máquinas de la red.
  - Todos los núcleos verifican si la dirección es la suya; si lo es, regresa un mensaje *aquí estoy* con su dirección en la red (número de máquina).
  - El núcleo emisor utiliza esa dirección y la captura para evitar a posteriori una nueva búsqueda del servidor.
- *Es un esquema transparente*, pero la transmisión provoca una carga adicional en el sistema:
  - Se puede evitar con una máquina adicional para la asociación de:
    - Los nombres de servicios.
    - Las direcciones de las máquinas.

Al utilizar este sistema:

- Se hace referencia a los procesos de los servidores mediante cadenas en ASCII que son las que aparecen en los programas.
- No se referencian números binarios de máquinas o procesos.
- Al ejecutar un cliente que intente utilizar un servidor:
  - En su primer intento envía una solicitud a un servidor especial de asociaciones (servidor de nombres):
    - Le solicita el número de la máquina donde en ese momento se localiza el servidor.
  - Conociendo la dirección del servidor, se le envía la solicitud del servicio requerido.

Otro método utiliza hardware especial:

- Los procesos eligen su dirección en forma aleatoria.
- Los chips de interfaz de la red se diseñan de modo que permitan a los procesos almacenar direcciones de procesos en ellos.
- Los paquetes transmitidos *utilizan direcciones de procesos* en vez de direcciones de máquinas.
- Al recibir cada paquete el chip de interfaz de la red debe examinarlo para determinar si el proceso destino se encuentra en esa máquina:
  - Lo acepta en caso afirmativo.
  - No lo acepta en caso negativo.

8.5 Primitivas de Bloqueo Vs. No Bloqueo en C - S

Las *primitivas de transferencia de mensajes* consideradas anteriormente se denominan *primitivas de bloqueo* o *primitivas síncronas* [25, Tanenbaum]:

- El proceso emisor se suspende (se bloquea) mientras se envía el mensaje.
- El proceso receptor se suspende mientras se recibe el mensaje.

Una alternativa son las *primitivas sin bloqueo* o *primitivas asíncronas:*

- El proceso emisor:
  - No se suspende mientras se envía el mensaje.
  - Sí puede continuar su cómputo paralelamente con la transmisión del mensaje.
  - No puede modificar el buffer de mensajes hasta que se envíe el mensaje.
  - No tiene control sobre la terminación de la transmisión y por lo tanto no sabe cuándo será seguro reutilizar el buffer.

Una solución es:

- Que el núcleo copie el mensaje a un buffer interno del núcleo.
- Que entonces el núcleo permita al proceso continuar y reutilizar el buffer.

La desventaja de la solución es que cada mensaje de salida debe ser copiado desde el espacio del usuario al espacio del núcleo.
Otra solución es:

- Interrumpir al emisor cuando se envíe el mensaje.
- Informarle que el buffer nuevamente está disponible.

La desventaja radica en la dificultad de la programación basada en interrupciones a nivel usuario.

Generalmente se considera que las desventajas de las primitivas asíncronas no compensan las ventajas del máximo paralelismo que permiten lograr.

El *criterio utilizado* ha sido el siguiente:

- La diferencia esencial entre una primitiva síncrona y una asíncrona es si el emisor puede volver a *utilizar el buffer de mensajes en forma inmediata y segura* después de recuperar el control.
- El momento en que el mensaje llega al receptor es irrelevante.

Otro criterio establece lo siguiente:

- Una primitiva síncrona es aquella en que el emisor se bloquea hasta que el receptor ha aceptado el mensaje y la confirmación regresa al emisor.
- Todo lo demás es asíncrono con este criterio.

Desde el *punto de vista del S. O.* generalmente se considera el primer criterio; el interés está centrado en el manejo de los buffers y en la transmisión de los mensajes.

Desde el *punto de vista de los lenguajes de programación* se tiende a considerar el segundo criterio; el interés está centrado en el lenguaje de programación y sus facilidades de uso.

Generalmente a las *primitivas de envío* se las conoce como **send** y a las de *recepción* como **receive** y ambas pueden ser *con bloqueo* o *sin bloqueo.*

Una *recepción sin bloqueo* le indica al núcleo la localización del buffer y regresa el control:

- El problema es saber quién hizo la llamada cuando se llevó a cabo la operación.
- Las soluciones pueden ser:
  - Proporcionar una primitiva explícita **wait** que permita al receptor bloquearse cuando lo desee.
  - Proporcionar una primitiva **test** que permita verificar el estado del núcleo.

8.6 Primitivas Almacenadas en Buffer Vs. No Almacenadas en C - S

Las primitivas consideradas hasta ahora son esencialmente *primitivas no almacenadas [25, Tanenbaum]:*

- *Significa que una dirección se refiere a un proceso específico.*
- Una llamada *receive (addr, &m)* le indica al núcleo de la máquina en donde se ejecuta:

- Que el proceso que hace la llamada escucha a la dirección *addr.*
- Que está preparada para recibir el mensaje enviado a esa dirección.
- Que se dispone de un único buffer de mensajes al que apunta m para capturar el mensaje que llegará.
- Que cuando el mensaje llegue será copiado (por el núcleo receptor) al buffer:
    - Se elimina entonces el bloqueo del proceso receptor.

Este esquema funciona bien cuando el servidor llama a *receive* antes de que el cliente llame a *send.*

El problema se presenta cuando el *send* se lleva a cabo antes que el *receive:*

- El núcleo del servidor:
    - No sabe cuál de sus procesos utiliza la dirección en el mensaje recién llegado.
    - No sabe dónde copiar el mensaje recibido.

Una solución consiste en:

- Descartar el mensaje.
- Dejar que el cliente espere.
- Confiar en que el servidor llame a *receive* antes de que el cliente vuelva a transmitir; por ello el cliente podría tener que intentar varias veces.

Si dos o más clientes utilizan un *servidor con transferencia de mensajes sin almacenamiento en buffers:*

- Luego de que el servidor aceptó un mensaje de uno de ellos:
    - Deja de escuchar a su dirección hasta que termina su trabajo.
    - Regresa al principio del ciclo para volver a llamar a *receive.*
- Si realizar el trabajo insume cierto tiempo, los demás clientes podrían hacer varios intentos de envíos sin éxito.

Otra solución consiste en hacer que *el núcleo receptor mantenga pendientes los mensajes* por un instante:

- Para prevenir que un *receive* adecuado se realice en un tiempo corto.
- Cuando llega un mensaje *"no deseado"*, se inicializa el cronómetro:
    - Si el tiempo expira antes de que ocurra un *receive* apropiado, el mensaje se descarta.
- Se reduce la probabilidad de que un mensaje se pierda.

- Se debe almacenar y manejar los mensajes que llegan en forma prematura.
- Se necesitan los buffers y la administración de los mismos.
- Se puede hacer mediante una nueva estructura de datos llamada **buzón:**
  - Un proceso interesado en recibir mensajes:
    - Le indica al núcleo que cree un *buzón* para él.
    - Especifica una dirección en la cual buscar los paquetes de la red.
  - Todos los mensajes que lleguen en esa dirección se colocan en el buzón.

La llamada a *receive* elimina un mensaje del *buzón* o se bloquea (si se utilizan primitivas con bloqueo) si no hay un mensaje presente.

Esta técnica se denomina *primitiva con almacenamiento en buffers.*

Los buzones tienen el problema de que *son finitos* y pueden ocuparse en su totalidad:

- Cuando llega un mensaje a un buzón lleno, el núcleo debe elegir entre:
  - Mantener el mensaje pendiente por un momento esperando que algún mensaje sea retirado del buzón a tiempo.
  - Descartar el mensaje.
- Esta es *la misma situación* que se tiene cuando se trabaja sin almacenamiento en buffers:
  - Con buffers se reduce la probabilidad de problemas, pero los problemas no se eliminan ni cambia su naturaleza.

Otra solución utilizada es *no dejar que un proceso envíe un mensaje si no existe espacio para su almacenamiento en el destino:*

- El emisor debe bloquearse hasta que obtenga de regreso un reconocimiento:
  - Debe indicar que el mensaje ha sido recibido.
- Si el buzón está lleno, el emisor puede hacer un respaldo y suspenderse de manera retroactiva:
  - La suspensión debe operar como si fuera *justo antes* de que intentara enviar el mensaje.
  - Cuando haya espacio libre en el buzón, se hará que el emisor intente nuevamente.

8.7 Primitivas Confiables Vs. No Confiables en C - S

Hasta acá se ha supuesto que *los mensajes enviados* ***siempre*** *serán recibidos [25, Tanenbaum]*, pero en la realidad, los mensajes *se pueden perder* por diversas causas.

Cuando un *cliente envía un mensaje* se le puede *suspender hasta que el mensaje ha sido enviado:*

- Cuando continúa, no hay garantía de que el mensaje ha sido entregado, pues el mensaje podría haberse perdido.

Un enfoque de este problema consiste en volver a definir la semántica de *send* para hacerlo *no confiable [23, Tanenbaum]:*

- El sistema *no garantiza* la entrega de los mensajes.
- La implantación de una comunicación confiable se deja en manos de los usuarios.

Otro método exige que *el núcleo de la máquina receptora envíe un reconocimiento al núcleo de la máquina emisora:*

- Solo cuando reciba el reconocimiento, el núcleo emisor liberará al proceso usuario (cliente).
- La solicitud de un cliente a un servidor es reconocida por el núcleo del servidor.
- La respuesta del servidor de regreso al cliente es reconocida por el núcleo del cliente.
- Una solicitud de respuesta consta de cuatro mensajes.

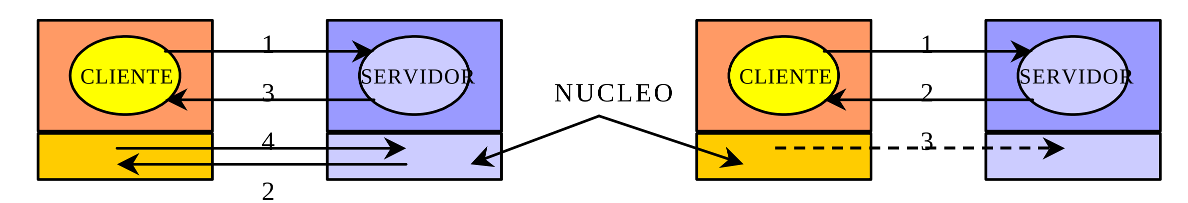
Otra solución aprovecha el hecho de que la *comunicación cliente - servidor se estructura:*

- Como una solicitud del cliente al servidor.
- Seguida de una respuesta del servidor al cliente.
- El cliente se bloquea después de enviar un mensaje.
- El núcleo del servidor no envía de regreso un reconocimiento sino que la misma respuesta funciona como tal.
- El emisor permanece bloqueado hasta que regresa la respuesta.
- Si la respuesta no llega en cierto tiempo, el núcleo emisor puede volver a enviar la solicitud; así se protege contra la posibilidad de una pérdida del mensaje.

En el esquema anterior *la respuesta funciona como un reconocimiento a la solicitud:*[4]

- No existe un reconocimiento por la respuesta:

[4] Ver Figura 8.4 de la página 255 [25, Tanenbaum].



1. SOLICITUD (DEL CLIENTE AL SERVIDOR).
2. RECONOCIMIENTO (DE NUCLEO A NUCLEO).
3. RESPUESTA (DEL SERVIDOR AL CLIENTE).
4. RECONOCIMIENTO (DE NUCLEO A NUCLEO).

1. SOLICITUD (DEL CLIENTE AL SERVIDOR).
2. RESPUESTA (DEL SERVIDOR AL CLIENTE).
3. RECONOCIMIENTO (DE NUCLEO A NUCLEO) - NO SIEMPRE EXISTE, POR ELLO APARECE COMO LINEA DISCONTINUA.

Figura 8.4: Esquema de mensajes reconocidos en forma individual y esquema donde la respuesta se utiliza como reconocimiento de la solicitud.

  - La seriedad de esta omisión depende de la naturaleza de la solicitud.

- En algunos casos se utiliza un reconocimiento del núcleo del cliente al núcleo del servidor:
  - Hasta no recibir este paquete, el *send* del servidor no termina y el servidor permanece bloqueado (si se utilizan primitivas con bloqueo).

Otra posibilidad es la siguiente:

- Al llegar una solicitud al núcleo del servidor, se inicia un cronómetro.
- Si el servidor envía la respuesta antes de que termine el cronómetro, ésta funciona como el reconocimiento.
- Si expira el tiempo del cronómetro, se envía un reconocimiento separado.

8.8 Implantación del Modelo C - S

Las principales *opciones de diseño* analizadas se resumen en [25, Tanenbaum]:

- *Direccionamiento:*
  - Número de máquina.
  - Direcciones ralas de procesos.
  - Búsqueda de nombres en ASCII por medio del servidor.
- *Bloqueo:*

  - Primitivas con bloqueo.
  - Sin bloqueo, con copia al núcleo.
  - Sin bloqueo, con interrupciones.

- *Almacenamiento en buffers:*
  - No usar el almacenamiento en buffers, descartar los mensajes inesperados.
  - Sin almacenamiento en buffers, mantenimiento temporal de los mensajes inesperados.
  - Buzones.

- *Confiabilidad:*
  - No confiable.
  - Solicitud - reconocimiento - respuesta - reconocimiento.
  - Solicitud - respuesta - reconocimiento.

Existen $3^4 = 81$ combinaciones, pero no todas son igual de buenas.
En el caso de *mensajes compuestos por varios paquetes*, el *reconocimiento* puede ser:

- *Por paquete individual:*
  - Ante la pérdida de un paquete, solo retransmite ése paquete.
  - Requiere *más paquetes* en la red.

- *Por mensaje completo:*
  - La recuperación es compleja ante la pérdida de un paquete.
  - Requiere *menos paquetes* en la red.

Otro aspecto interesante de la implementación es el protocolo subyacente utilizado en la comunicación c - s.
Los *principales tipos de paquetes* son los siguientes:

- **Req:**
  - Solicitud.
  - De cliente a servidor.
  - El cliente desea servicio.

- **Rep:**
  - Respuesta.
  - De servidor a cliente.
  - Respuesta del servidor al cliente.

- **Ack:**
  - Reconocimiento.
  - De cualquiera de ellos a algún otro.
  - El paquete anterior que ha llegado.
- **Aya:**
  - ¿Estás vivo?.
  - De cliente a servidor.
  - Verifica si el servidor se ha descompuesto.
- **Iaa:**
  - Estoy vivo.
  - De servidor a cliente.
  - El servidor no se ha descompuesto.
- **Ta:**
  - Intenta de nuevo.
  - De servidor a clientes.
  - El servidor no tiene espacio.
- **Au:**
  - Dirección desconocida.
  - De servidor a cliente.
  - Ningún proceso utiliza esta dirección.

Algunos ejemplos de intercambio de paquetes para la comunicación cliente - servidor pueden verse en la Figura 8.5 de la página 258 [25, Tanenbaum].

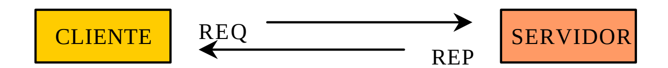
8.9 Llamada a un Procedimiento Remoto (RPC)

El *modelo cliente - servidor* es una forma conveniente de estructurar un S. O. distribuido, pero *posee una falencia [25, Tanenbaum]:*

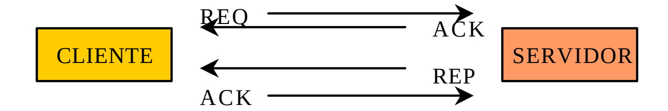
- El *paradigma esencial* en torno al que se construye la comunicación es la *entrada / salida.*
- Los *procedimientos send / receive* están reservados para la realización de *e / s.*

Una opción distinta fue planteada por **Birrel** y **Nelson:**

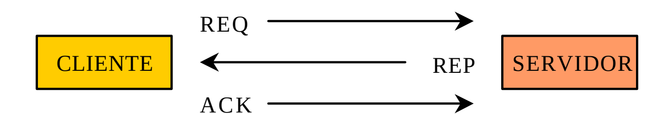
SOLICITUD / RESPUESTA DIRECTA, SIN RECONOCIMIENTOS



RECONOCIMIENTO DE CADA MENSAJE INDIVIDUAL



LA RESPUESTA ACTUA COMO RECONOCIMIENTO



VERIFICACION DE UN CLIENTE RESPECTO DE SI EL SERVIDOR ESTA ACTIVO

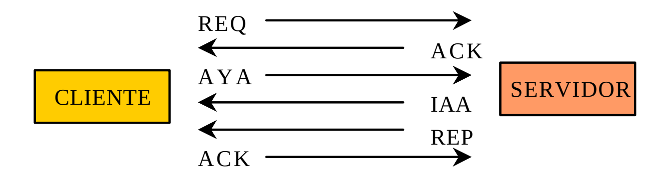


Figura 8.5: Intercambio de paquetes en la comunicación cliente - servidor.

- Permitir a los programas que llamasen a *procedimientos localizados en otras máquinas.*
- Cuando un proceso en la máquina *"A"* llama a un procedimiento en la máquina *"B":*
  - El proceso que realiza la llamada se suspende.
  - La ejecución del procedimiento se realiza en *"B".*
- La información se puede transportar de un lado al otro mediante los parámetros y puede regresar en el resultado del procedimiento.
- El programador no se preocupa de una transferencia de mensajes o de la e / s.
- A este método se lo denomina **llamada a procedimiento remoto** o **RPC**.
- El procedimiento que hace la llamada y el que la recibe se ejecutan en máquinas diferentes, es decir que utilizan espacios de direcciones distintos.

8.10 Operación Básica de RPC

Una *llamada convencional a un procedimiento*, es decir en una sola máquina, funciona de la siguiente manera [25, Tanenbaum]:[5]

- Sea *count = read (fd, buf, nbytes)*; donde:
  - *fd* es un entero; *buf* es un arreglo de caracteres; *nbytes* es otro entero.
- El programa llamador coloca los parámetros en la pila.
- El procedimiento llamado desde el programa llamador se carga en la memoria.
- Después de que *read* termina su ejecución:
  - Coloca el valor de regreso en un registro.
  - Elimina la dirección de regreso.
  - Transfiere de nuevo el control a quien hizo la llamada.
  - Quien hizo la llamada elimina los parámetros de la pila y regresa a su estado original.

Los *parámetros* pueden llamarse **por valor** o **por referencia.**
Un parámetro por valor:

- Se copia a la pila.
- Para el procedimiento que recibe la llamada es solo una variable local ya inicializada.

[5] Ver Figura 8.6 de la página 260 [25, Tanenbaum].

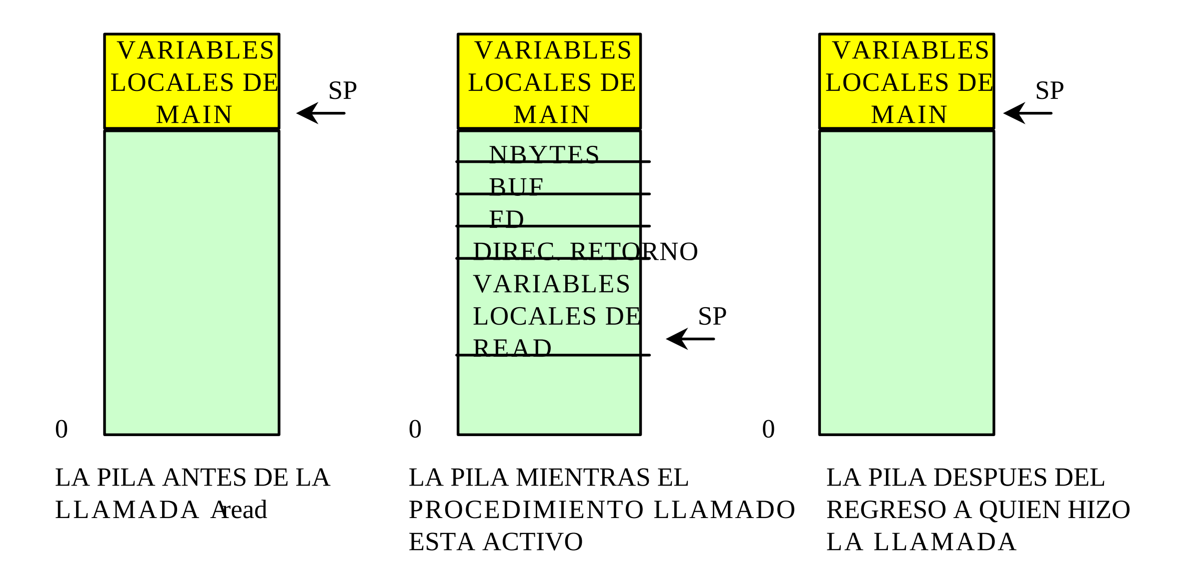


Figura 8.6: Llamada a un procedimiento local.

- El procedimiento podría modificarla, sin que esto afecte el valor de la variable original en el procedimiento que hizo la llamada.

Un parámetro por referencia:

- Es un apuntador a una variable (es decir, la dirección de la variable), no el valor de la variable.
- En el ej. anterior, válido para "C", el segundo parámetro es un parámetro por referencia y es un arreglo.
- Si el procedimiento que recibe la llamada utiliza este parámetro por referencia para almacenar algo en el arreglo, modifica el arreglo en el procedimiento que hizo la llamada

Otro mecanismo para el paso de parámetros es la llamada por copiar / restaurar:

- Quien recibe la llamada copia la variable en la pila, como en la llamada por valor.
- La copia de nuevo después de la llamada, escribiendo sobre el valor original.

La decisión de cuál mecanismo utilizar para el paso de parámetros la toman los diseñadores del sistema y es una propiedad fija del lenguaje.

La ida es que una llamada a un procedimiento remoto (RPC) se parezca lo más posible a una llamada local:

- La RPC debe ser transparente.
- El procedimiento que hace la llamada no debe ser consciente de que el procedimiento llamado se ejecuta en una máquina distinta, o viceversa.

- Si *read* es un procedimiento remoto (ej.: se ejecuta en la máquina del servidor de archivos) se coloca en la biblioteca una versión distinta de *read* llamada *stub del cliente:*
  - No coloca los parámetros en registros y le pide al núcleo que le proporcione datos.
  - Coloca los parámetros en un mensaje y le pide la núcleo que envíe el mensaje al servidor.
  - Después de la llamada a *send*, el *stub del cliente* llama a *receive* y se bloquea hasta que regrese la respuesta.

Cuando el mensaje llega al *servidor:*

- El núcleo lo transfiere a un *stub del servidor.*
- Generalmente el *stub del servidor* ya habrá llamado a *receive* y estará bloqueado esperando que le lleguen mensajes.
- El *stub del servidor:*
  - "desempaca" los parámetros del mensaje.
  - Llama al procedimiento del servidor de la manera convencional.

Para el *servidor* es como si tuviera una llamada directa del cliente; lleva a cabo el trabajo y regresa el resultado a quien hizo la llamada, de la forma usual.

El *stub del servidor* recupera el control luego de la llamada y:

- Empaca el resultado en un mensaje.
- Llama a *send* para regresarlo al cliente.
- Llama a *receive* y espera el siguiente mensaje.

Cuando el mensaje regresa a la *máquina cliente:*

- El núcleo ve que está dirigido al proceso cliente.
- El mensaje se copia al buffer en espera.
- El proceso cliente elimina su bloqueo.
- El *stub del cliente* examina el mensaje, desempaca el resultado, lo copia a quien hizo la llamada y regresa de la manera usual.

Cuando el *proceso que hizo la llamada* obtiene el control luego de la llamada a *read:*

- Dispone de los datos.
- Ignora que el trabajo se realizó de manera remota.

- Ha tenido acceso a servicios remotos mediante llamadas comunes a procedimientos locales.

Resumiendo, se pueden indicar los siguientes *pasos como una llamada a un procedimiento remoto:*[6]

- El *procedimiento cliente* llama al *stub del cliente* de la manera usual.
- El *stub del cliente* construye un *mensaje* y hace un señalamiento al núcleo.
- El núcleo envía el *mensaje* al núcleo remoto.
- El núcleo remoto proporciona el *mensaje* al *stub del servidor.*
- El *stub del servidor* desempaca los parámetros y llama al *servidor.*
- El *servidor* realiza el trabajo y regresa el resultado al *stub.*
- El *stub del servidor* empaca el resultado en un *mensaje* y hace un señalamiento al núcleo.
- El núcleo remoto envía el *mensaje* al núcleo del cliente.
- El núcleo del cliente da el mensaje al *stub del cliente.*
- El *stub* desempaca el resultado y regresa al cliente.

Se ha convertido la llamada local del procedimiento cliente al stub del cliente, en una llamada local al procedimiento servidor.

8.11 Transferencia de Parámetros en RPC

El *empacamiento de parámetros en un mensaje* se llama *ordenamiento de parámetros [25, Tanenbaum].*

El *mensaje* también contiene el nombre o número del procedimiento por llamar; el servidor podría soportar varias llamadas y se el tiene que indicar cuál de ellas se necesita.

Cuando el *mensaje llega al servidor:*

- El *resguardo (stub)* lo examina para ver cuál procedimiento necesita.
- Lleva a cabo la llamada apropiada.

La *llamada real del resguardo al servidor* es similar a la *llamada original del cliente*, excepto en que *los parámetros son variables inicializadas a partir del mensaje recibido*, en vez de ser constantes.[7]

Los *elementos del mensaje* corresponden a:

- Identificador del procedimiento.

[6] Ver Figura 8.7 de la página 263 [25, Tanenbaum].
[7] Ver Figura 8.8 de la página 263 [25, Tanenbaum].

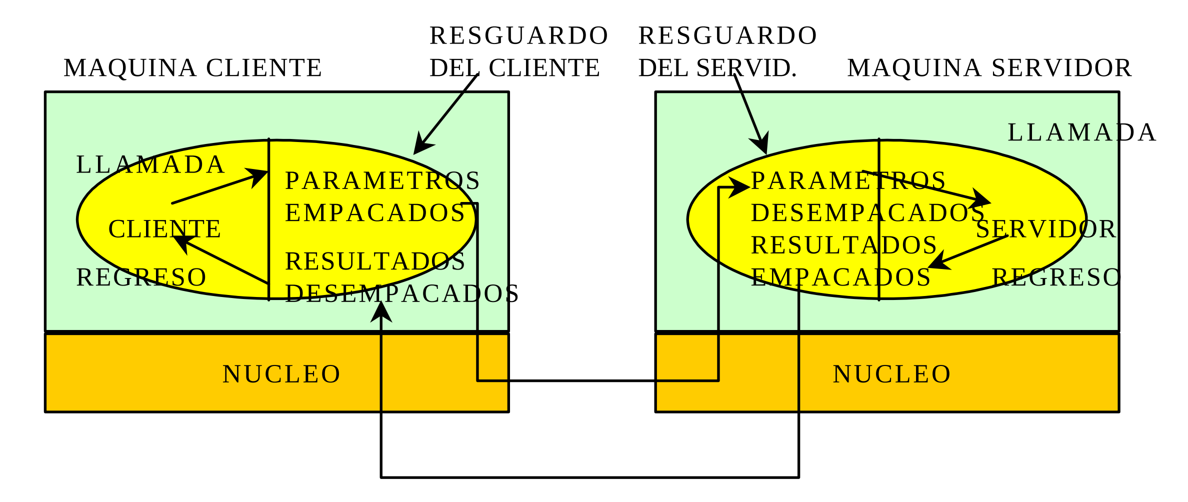


Figura 8.7: Llamadas y mensajes en una RPC, donde cada elipse representa un solo proceso que incluye el resguardo.

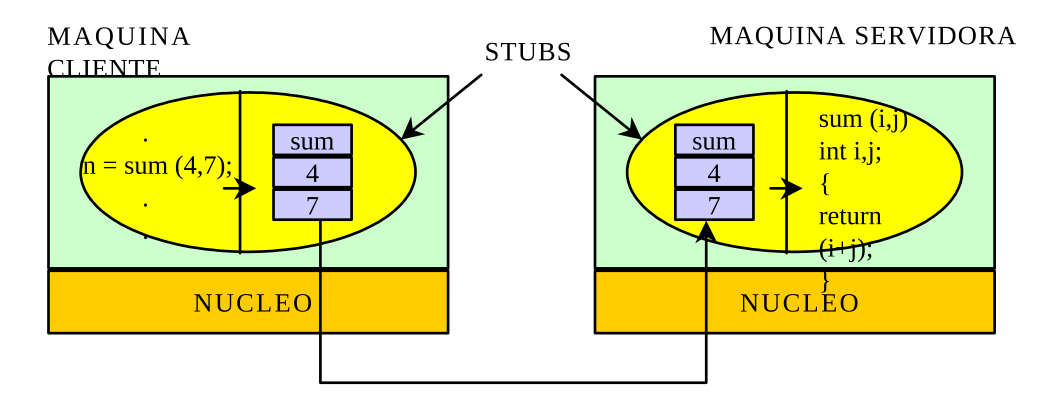


Figura 8.8: Ejemplo de cálculo remoto.

- Parámetros.

Un *mensaje* que corresponda a un procedimiento remoto con *"n"* parámetros tendrá *"n + 1"* campos:

- Uno para identificar al procedimiento.
- Uno para cada uno de los *"n"* parámetros.

Un aspecto importante es determinar cómo se debe *representar la información en los mensajes:*

- Una forma es:
  - Diseñar un estándar de red o *forma canónica* para los enteros, caracteres, booleanos, número de punto flotante, etc.
  - Pedir a todos los emisores que conviertan sus representaciones internas a esta forma durante el ordenamiento.

Es importante saber la *organización del mensaje* y la *identidad del cliente.*
También es importante determinar *de dónde provienen los procedimientos resguardo:*

- En muchos sistemas se generan automáticamente.
- Dada una especificación del procedimiento servidor y las reglas de codificación, el formato del mensaje queda determinado de manera única.

Es posible tener un *compilador* que:

- Lea las especificaciones del servidor.
- Genere un resguardo del cliente que empaque sus parámetros en el formato estándar de los mensajes.
- Genere un resguardo del servidor que los desempaque.
- Llame al servidor.

Un aspecto también muy importante es *cómo se transfieren los apuntadores*, pues un apuntador solo tiene sentido dentro del espacio de direcciones del proceso en el que se utiliza.

Una solución es *copiar en el mensaje la estructura* para la cual se utiliza el apuntador y enviarlo al servidor, en tal caso los cambios que realice el servidor mediante el apuntador afectarán directamente al buffer de mensajes en el resguardo del servidor.

8.12 Conexión Dinámica (Dynamic Binding) en RPC

Un tema fundamental es la *forma en que el cliente localiza al servidor [25, Tanenbaum].*

Un método consiste en *integrar dentro del código del cliente* la dirección (en la red) del servidor:

- El problema es que resulta demasiado rígido.
- Si el servidor se desplaza, si se duplica o si cambia la interfaz, habría que localizar y volver a compilar los numerosos programas.

Una solución es la **conexión dinámica** para que concuerden los clientes y los servidores.

El punto de inicio de la conexión dinámica es la *especificación formal del servidor*, que indica el nombre del servidor, el número de versión y una lista de los procedimientos que proporciona.

Se tienen los *tipos de parámetros* para cada procedimiento y cada parámetro queda determinado como parámetro *in, out* o *in-out.*

La dirección es relativa al servidor.

El principal uso de la especificación formal es como entrada del *generador de resguardos:*

- Produce el resguardo del cliente y el del servidor.
- Ambos resguardos se colocan en las bibliotecas respectivas.

Cuando un *programa (cliente)* llama a cualquiera de los procedimientos definidos mediante esa especificación, el correspondiente *procedimiento resguardo del cliente* se liga con su binario.

Si se compila un *programa servidor*, los *resguardos del servidor* se le ligan también.

Cuando el *servidor* inicia su ejecución:

- Una llamada tipo *initialize* que se encuentra fuera del ciclo principal *exporta la interfaz del servidor:*
  - El servidor envía un mensaje a un programa *conector* para darle a conocer su existencia.
  - Esto es el *registro del servidor (registering the server).*
  - El servidor proporciona al conector su nombre, número de versión y un único *identificador.*
  - El identificador generalmente tiene una longitud de 32 bits y un *asa (handle)* que se utiliza para localizarlo.
  - El *asa (handle)* depende del sistema y puede ser:
    * Una dirección ethernet, ip, x.500.
    * Un identificador ralo de procesos, etc.
  - El asa también puede proporcionar información relativa a la autentificación.

Un *servidor* puede *cancelar su registro* con el conector si ya no está preparado para prestar algún servicio.

El *cliente localiza al servidor* de la siguiente manera:

- Cuando el cliente llama a alguno de los procedimientos remotos por primera vez:
  - El resguardo del cliente:
    - Ve que aún no está conectado a un servidor.
    - Envía un mensaje al *conector* solicitando la *importación* de cierta versión de cierta interfaz.
  - El conector verifica si uno o más servidores ya han exportado una interfaz con ese nombre y versión.
  - Si ninguno de los servidores en ejecución en ese momento soporta esa interfaz, la llamada fracasa.
  - Si existe un servidor adecuado, el conector proporciona un asa e identificador único al resguardo del cliente, que utiliza el asa como la dirección a la cual enviar el mensaje solicitado.

Es un *esquema muy flexible* pero el *conector puede ser un cuello de botella* con altas cargas de trabajo.

8.13 Semántica de RPC en Presencia de Fallos

El objetivo de RPC es ocultar la comunicación al hacer que las llamadas a procedimientos remotos se parezcan a las llamadas locales [25, Tanenbaum].

El problema se presenta cuando aparecen los *errores*, ya que las diferencias entre las llamadas locales y remotas no son tan fáciles de encubrir.

Se consideraran las siguientes *situaciones:*

- *El cliente no puede localizar al servidor.*
- *Se pierde el mensaje de solicitud del cliente al servidor.*
- *Se pierde el mensaje de respuesta del servidor al cliente.*
- *El servidor falla antes de recibir una solicitud.*
- *El cliente falla después de enviar una solicitud.*

8.13.1 El Cliente No Puede Localizar al Servidor

El servidor podría estar inactivo.

El servidor podría estar utilizando una nueva versión de la interfaz y nuevos resguardos, que no serían compatibles con la interfaz y los resguardos del cliente.

En el servidor, cada uno de los procedimientos regresa un valor:

- Generalmente el código -1 indica un fallo.

- También se suele utilizar una variable global (UNIX) *errno* a la que se asigna un valor que indica el tipo de error.

- Un tipo de error sería "no se pudo localizar al servidor".

Otra posibilidad para el tratamiento de los errores es mediante una *excepción* provocada por el error:

- Se codifican procedimientos especiales que son llamados ante errores específicos.

- El problema es que se puede destruir la transparencia deseada, ya que se dificulta mantener la similitud entre procedimientos locales y remotos.

8.13.2 Pérdida de Mensajes de Solicitud

El núcleo (kernel) debe *inicializar un cronómetro al enviar la solicitud*; si el tiempo se termina antes de que regrese una respuesta o reconocimiento, el núcleo *vuelve a enviar el mensaje.*

Si el mensaje *realmente se perdió*, el servidor no podrá indicar la diferencia entre la retransmisión y el original y todo funcionará bien.

Si el número de mensajes perdidos supera cierto *límite*, el núcleo puede asumir que el servidor está inactivo y se regresa a la situación "no se pudo localizar al servidor".

8.13.3 Pérdida de Mensajes de Respuesta

La pérdida de respuestas *genera mayores problemas que la pérdida de solicitudes.*

Se utiliza un cronómetro:

- Si no llega una respuesta en un período razonable, se debe volver a enviar la solicitud.

- El problema es que el núcleo del cliente no está seguro de la razón por la que no hubo respuesta.

Ciertas operaciones se pueden *repetir con seguridad* tantas veces como sea necesario sin que ocurran daños; una solicitud con esta propiedad es *idempotente.*

Otras operaciones no son idempotentes, por ej. la transferencia de dinero:

- Se emite una solicitud a un servidor bancario para transferir cierta suma de dinero.

- La solicitud llega y se efectúa pero se pierde la respuesta.

- El cliente considera que la solicitud se perdió y la emite nuevamente.

- El servidor recibe la nueva solicitud y la ejecuta al no saber que es un reenvío de la anterior.

Una *forma de resolver el problema* consiste en lo siguiente:

- El núcleo del cliente asigna a cada solicitud un número secuencial.

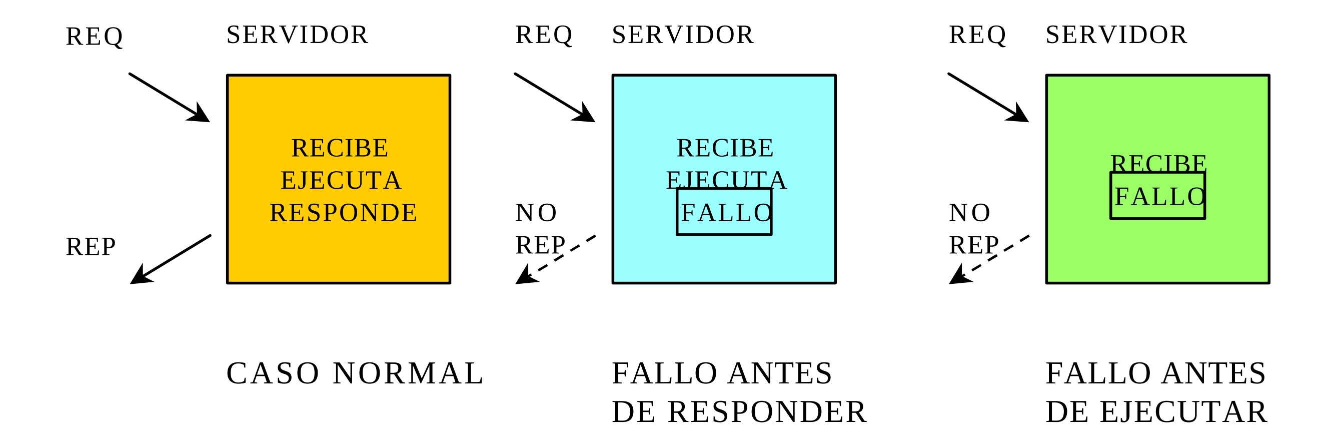


Figura 8.9: Situaciones posibles de fallos en el servidor.

- El núcleo del servidor mantiene un registro del número secuencial de recepción más reciente de cada uno de los núcleos de clientes que lo utilicen.
- El núcleo del servidor podrá indicar la *diferencia entre una solicitud original y una retransmisión* y puede rechazar la realización de cualquier solicitud por segunda vez.

Una *protección adicional* es tener un bit en el encabezado del mensaje para distinguir las solicitudes de las retransmisiones.

8.13.4 Fallos del Servidor

Un fallo del servidor *también se relaciona con la idempotencia* pero no se puede resolver con números secuenciales.[8]

El problema es que *el núcleo del cliente no puede decidir* si se ha presentado la segunda o la tercera situación.

Las *posibles soluciones* son las siguientes:

- *Semántica al menos una:*
  - Esperar hasta que el servidor vuelva a arrancar (o se reconecte a un nuevo servidor) e intente realizar de nuevo la operación.
  - Mantener el intento hasta recibir una respuesta para dársela al cliente.
  - Garantiza que la RPC se ha realizado *al menos una vez*, pero es posible que se realice más veces.
- *Semántica a lo más una:*
  - No se reintenta y se informa del fallo.

[8] Ver Figura 8.9 de la página 268 [25, Tanenbaum].

  - Garantiza que la RPC se realiza *a lo más una vez*, pero es posible que no se realice ni una sola vez.

- *Semántica de no garantizar nada:*
  - Cuando un servidor falla, el cliente no obtiene ayuda o alguna promesa.
  - La RPC se puede realizar en cualquier lugar, un número de veces que va *desde "0" hasta "n".*
  - Resulta fácil de implantar.

- *Semántica de exactamente una:*
  - Es la *solución deseable* pero generalmente *no* existe forma de garantizar esto.
  - El procedimiento de recuperación depende totalmente del momento en que ocurre el fallo.
  - El cliente no tiene forma de descubrir ese instante.

La posibilidad de *fallos del servidor* distingue de manera clara los *sistemas con un único procesador* de los *sistemas distribuidos:*

- Con un único procesador el fallo de un servidor implica un fallo del cliente y la recuperación no es ni posible ni necesaria.
- Con sistemas distribuidos es posible y necesario realizar cierta acción.

8.13.5 Fallos del Cliente

La cuestión es qué ocurre si un cliente envía una solicitud a un servidor y *falla antes de que el servidor responda.*

Se genera una solicitud de trabajo o cómputo que al fallar el cliente ya nadie espera; se dice que se tiene un *cómputo huérfano.*

Los *principales problemas* generados por cómputos huérfanos son los siguientes:

- Desperdicio de ciclos de cpu.
- Posible bloqueo de archivos.
- Apropiación de recursos valiosos.
- Posible confusión cuando:
  - El cliente rearranca y efectúa de nuevo la RPC.
  - La respuesta del huérfano regresa inmediatamente luego.

Las *soluciones a los cómputos huerfanos* son las siguientes [25, Tanenbaum]:

- *Exterminación:*

- Se crea un registro que indica lo que va a hacer el resguardo del cliente antes de que emita la RPC.
- El registro se mantiene en disco.
- Luego del rearranque se verifica el contenido del registro y se elimina el huérfano explícitamente.
- La desventaja es la sobrecarga en e / s generada por la grabación previa a cada RPC.
- Fallaría si los huérfanos generan RPC, creando *huérfanos de huérfanos:*
  - Sería imposible localizarlos.
  - Ante ciertos fallos en la red sería imposible eliminarlos aunque se los localice.

- *Reencarnación:*
  - Resuelve los problemas anteriores sin necesidad de escribir registros en disco.
  - Consiste en dividir el tiempo en *épocas* numeradas de manera secuencial.
  - Cuando un cliente rearranca envía un mensaje a todas las máquinas declarando el inicio de una nueva época.
  - Al recibirse estos mensajes se eliminan todos los cómputos remotos.
  - Si se divide la red mediante particiones por fallas, podrían sobrevivir ciertos huérfanos:
    - Cuando se reconecten y vuelvan a reportarse sus respuestas contendrán un número de época obsoleto y se los podrá detectar y eliminar.

- *Reencarnación sutil:*
  - Cuando llega un mensaje de cierta época:
    - Cada máquina verifica si tiene cómputos remotos:
      - En caso afirmativo intenta localizar a su poseedor.
      - Si no se localiza al poseedor se elimina el cómputo.

- *Expiración:*
  - A cada RPC se le asigna una cantidad estándar de tiempo *"t"* para que realice su trabajo.
  - Si el tiempo es insuficiente debe pedir explícitamente otro quantum, pero esto es un inconveniente.
  - Si luego del fallo el servidor espera *"t"* antes de rearrancar, todos los huérfanos habrán desaparecido.
  - El problema es elegir un *"t"* razonable, ya que pueden existir RPC con requisitos diversos.

8.14 Aspectos de la Implantación en RPC

El *desempeño o performance* es fundamental en los sistemas distribuidos [25, Tanenbaum].

El *desempeño* depende de manera crítica de la *velocidad de la comunicación.*

La *velocidad* depende en gran medida de la *implantación.*

8.14.1 Protocolos RPC

Se debe elegir entre un *protocolo orientado a la conexión* o un *protocolo sin conexión.*

En los *protocolos orientados a la conexión:*

- Se establece una conexión entre cliente y servidor.
- Todo el tráfico en ambas direcciones utiliza esa conexión.
- Se maneja a un nivel inferior mediante el software que soporta la conexión.
- Es muy útil para *redes de área amplia o extendida (WAN).*
- Es desventajoso en *redes de área local (LAN):*
  - Por la pérdida de performance que significaría procesar software adicional para aprovechar la ventaja de no perder los paquetes; esto difícilmente se precisa en las LAN.
- Muchos sistemas distribuidos en *áreas geográficas reducidas* utilizan *protocolos sin conexión.*

Los *protocolos sin conexión* tienen características opuestas a las indicadas precedentemente.

Otra opción importante es utilizar un *protocolo estándar* de propósito general o alguno *específico para RPC.*

La utilización del *protocolo estándar* **IP** (o **UDP**, integrado a IP) posee las siguientes *ventajas:*

- El protocolo ya fue diseñado, lo que ahorra trabajo.
- Se dispone de muchas implantaciones, lo que también ahorra trabajo.
- Los paquetes IP se pueden enviar y recibir por casi todos los sistemas UNIX.
- Los paquetes IP y UDP se pueden transmitir en muchas de las redes existentes.

El *problema con un protocolo estándar* tipo *IP* es el desempeño o *performance:*

- IP no se diseñó como un protocolo de usuario final.
- IP se diseñó como una base para que las conexiones confiables **TCP** se pudiesen establecer en las interredes.

- IP soporta la fragmentación de paquetes para adecuarlos a redes con un pequeño tamaño máximo de paquete.

- Esta característica de fragmentación no se necesita en un sistema distribuido basado en una LAN, pero:

  - Los campos del encabezado del paquete IP relacionados con la fragmentación deben ser:
    - Llenados por el emisor.
    - Verificados por el receptor.
  - Los paquetes IP tienen 13 campos de encabezado:
    - Son útiles solo 3:
      - Dirección fuente.
      - Dirección destino.
      - Longitud del paquete.
    - Los 10 campos restantes incluyen uno de suma de verificación, cuyo cálculo consume cpu.
    - UDP tiene otra suma de verificación, que también cubre los datos y consume cpu.

La alternativa es utilizar un *protocolo especializado en RPC*, que debe ser inventado, implantado, probado e insertado en los sistemas existentes:

- Debe ser de alto rendimiento.

- Debe ser aceptado masivamente.

Otro aspecto importante relacionado con los protocolos es la *longitud del paquete y el mensaje:*

- La realización de una RPC tiene un alto costo fijo independiente de la cantidad de datos enviados.

- El protocolo y la red deben permitir transmisiones largas para *minimizar* el número de RPC.

8.14.2 Reconocimientos

Cuando los RPC de gran tamaño deben dividirse en *muchos paquetes pequeños*, surge la cuestión del *reconocimiento:*

- Los paquetes serán *reconocidos grupalmente o individualmente.*

- Ej.: un cliente desea escribir un bloque de datos de 4k en un servidor de archivos, pero el sistema no puede manejar paquetes mayores de 1k.

Una *estrategia de reconocimiento* es el *protocolo detenerse y esperar (Stop-And-Wait Protocol):*

- El cliente envía el paquete 0 con el primer 1k.
- El cliente espera un reconocimiento del servidor.
- El cliente envía el paquete 1 con el segundo 1k.
- Etc.
- La pérdida de un paquete significa que no llegará su reconocimiento y habrá que retransmitirlo.

Otra estrategia es el *protocolo de chorro (Blast Protocol):*

- El cliente envía todos los paquetes tan pronto como puede.
- El servidor reconoce todo el mensaje al recibir todos los paquetes; no hay reconocimiento individual de paquetes.
- La pérdida de un paquete puede significar:
  - La retransmisión de todo el mensaje.
  - La *repetición selectiva* de la transmisión del paquete dañado o perdido.

Otra consideración *más importante que el control de errores es el control del flujo:*

- Está relacionado con la capacidad finita de recepción de paquetes adyacentes por parte de los chips de interfaz de red.
- Cuando un paquete llega a un receptor que no lo puede aceptar se presenta un *error de sobreejecución (overrun error)* y el paquete se pierde.

Con el *protocolo detenerse y esperar* no se presentan los errores de sobreejecución.
Con el *protocolo de chorro* pueden ocurrir errores de sobreejecución.
Una solución consiste en que el emisor inserte un *retraso* entre los paquetes para darle tiempo al receptor para:

- Generar la interrupción correspondiente al paquete.
- Volver a estar listo para recepción.

Si la sobreejecución se debe a la capacidad finita del buffer en el chip de la red, el emisor puede:

- Enviar *"n"* paquetes y luego hacer una pausa.
- Solicitar una confirmación o reconocimiento luego de cada *"n"* paquetes.

Un *problema adicional* consiste en la *posible pérdida de paquetes de reconocimiento del cliente al servidor:*

- Para el cliente, que recibió la respuesta a su requerimiento, todo habrá terminado correctamente.
- Para el servidor habrá una respuesta no reconocida.
- Una solución es reconocer también a los paquetes de reconocimiento, lo cual agrega complejidad y costo adicional.
- Otra solución es que el servidor inicialice un cronómetro al enviar la respuesta:
  - La respuesta se descartará cuando ocurra alguna de las siguientes situaciones:
    * Llegada del reconocimiento.
    * Expiración del tiempo.
    * Llegada de un nuevo requerimiento del cliente.

8.14.3 Ruta Crítica

Es la serie de instrucciones que se ejecutan con cada RPC.[9]

Es importante determinar en *qué parte* de la ruta crítica se ocupa la *mayor parte del tiempo* que demanda la RPC:

- Depende del tipo de RPC y de la cantidad de datos que se deben transportar.
- En RPC con *transporte mínimo* la mayor parte del tiempo se ocupa en:
  - El cambio de contexto al resguardo del servidor al llegar un paquete.
  - La rutina de servicio de interrupciones.
  - El movimiento del paquete a la interfaz de la red para su transmisión.
- En RPC con *transporte de 1k o más* la mayor parte del tiempo se ocupa en:
  - El tiempo de transmisión.
  - El tiempo que tarda el desplazamiento del paquete hacia adentro y afuera de la interfaz.

8.14.4 Copiado

Un aspecto que domina frecuentemente los tiempos de ejecución en RPC es el *copiado.*

El número de veces que se debe copiar un *mensaje* varía según el hardware, el software y el tipo de llamada.

En el mejor de los casos el chip de la red puede utilizar el *DMA (acceso directo a la memoria)* para:

- Transferir el mensaje del espacio de direcciones del resguardo del cliente a la red.

[9] Ver Figura 8.10 de la página 275 [25, Tanenbaum].

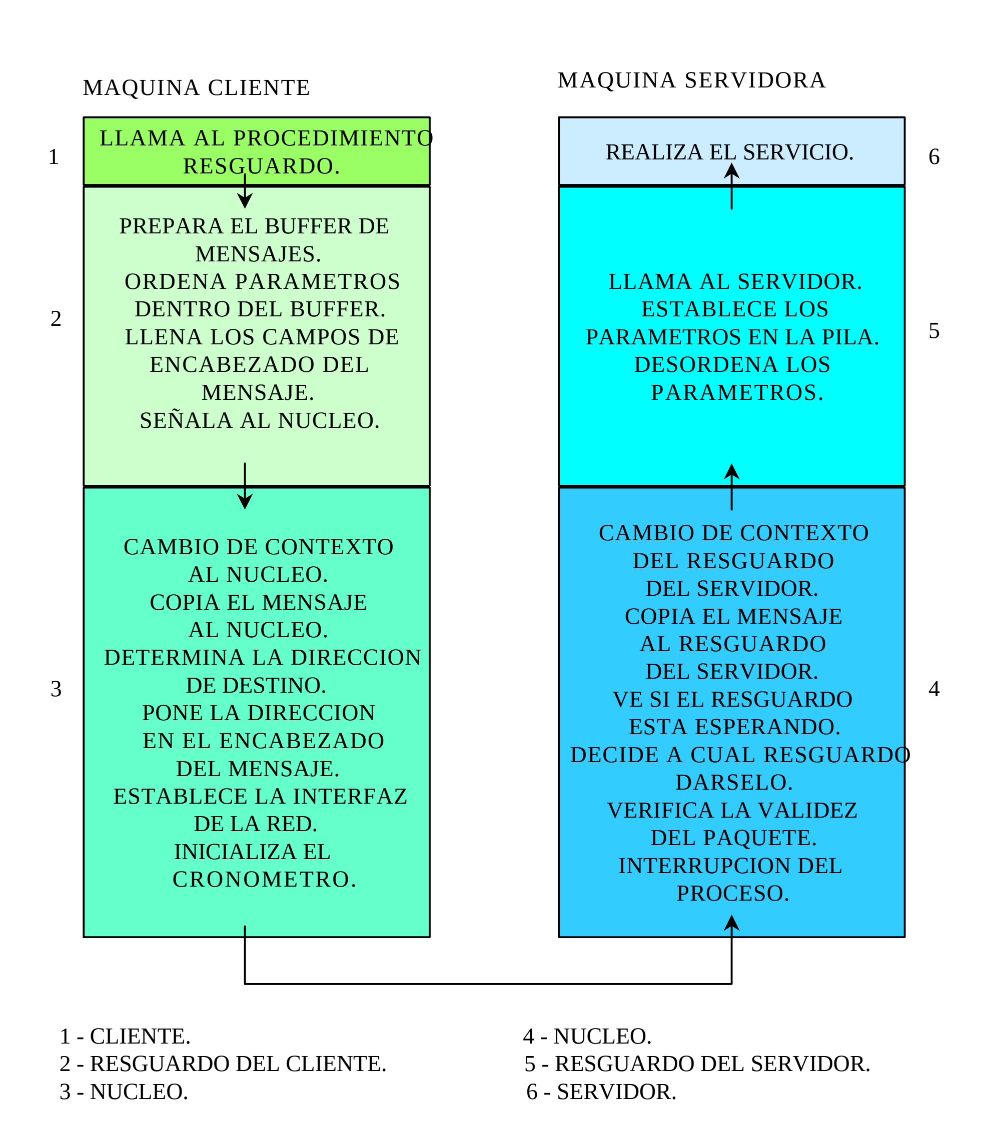


Figura 8.10: Ruta crítica del cliente al servidor.

- Depositarlo en la memoria del núcleo del servidor en tiempo real.
- El núcleo:
  - Inspecciona el paquete.
  - Asocia la página que lo contiene en el espacio de direcciones del servidor.
  - Si no puede efectuar la asociación, copia el paquete al resguardo del servidor.

Generalmente las *copias efectuadas* son las siguientes:

- El núcleo del cliente copia el mensaje del resguardo del cliente en un buffer del núcleo para su transmisión.
- El núcleo copia el mensaje, en software, a un buffer de hardware en la tarjeta de interfaz de la red.
- El paquete se desplaza por la red hacia la tarjeta de interfaz de la máquina destino.
- Cuando la interrupción correspondiente al paquete aparece en la máquina del servidor, el núcleo lo copia a un buffer del núcleo.
- El núcleo del servidor lo extrae del buffer de hardware.
- El mensaje, luego de ser inspeccionado, se copia al resguardo del servidor.
- Si la llamada transfiere como parámetro un arreglo de gran tamaño se agregan las siguientes copias:
  - A la pila del cliente para la llamada del resguardo.
  - De la pila al buffer de mensajes durante el ordenamiento dentro del resguardo del cliente.
  - Del mensaje recibido en el resguardo del servidor a la pila del servidor que antecede a la llamada al servidor.

El esquema precedente incluye 8 copias:

- Si el tiempo de copia de una palabra de 32 bits es de 500 nseg, con 8 copias, cada palabra necesita 4 microseg.
- Limita la velocidad máxima de transmisión de los datos a 1 mbyte / seg, independientemente de la velocidad de la red.

Una característica del *hardware* que disminuye el copiado es la *dispersión - asociación (scatter - gather):*

- El chip de la red organiza un paquete concatenando 2 o más buffers de memoria situados en distinto ámbito del cliente: núcleo, resguardo del cliente, etc.
- En el servidor ocurre algo similar, pero con más limitaciones.

En los S. O. con *memoria virtual* se puede evitar el copiado al resguardo mediante el siguiente procedimiento:

- El núcleo modifica el mapa de la memoria para asociar el buffer con el paquete en el espacio de direcciones del servidor y enviar simultáneamente el buffer del resguardo del servidor al núcleo.

- Los requisitos son los siguientes:

  - El buffer del paquete en el núcleo ocupa toda una página a partir de una frontera de página.
  - El buffer receptor del resguardo del servidor también es de toda una página e inicia en una frontera de página.

8.14.5 Manejo del Cronómetro

La mayoría de los protocolos inicializan un *cronómetro* cada vez que se envía un mensaje y se espera una respuesta.

Si la respuesta no llega en el tiempo esperado se vuelve a transmitir el *mensaje.*

Este proceso se puede repetir hasta un máximo previsto.

El tiempo de cpu destinado al manejo de los cronómetros puede ser considerable.

El establecimiento de un cronómetro requiere construir una *estructura de datos* que:

- Especifique el momento en que el cronómetro debe detenerse.

- La acción a realizar cuando eso suceda.

La estructura de datos de un cronómetro:

- Se inserta en una *lista de cronómetros pendientes* cuando se inicializa el cronómetro.

- Se retira de la lista cuando llega un reconocimiento antes de que termine el tiempo acordado.

Un valor de *lapso de expiración muy pequeño* hará que:

- Los cronómetros expiren con mucha frecuencia.

- Se realicen muchas retransmisiones innecesarias.

Un valor de *lapso de expiración muy grande* hará que:

- Se demore en retransmitir un paquete realmente perdido.

Una *alternativa* al almacenamiento de los cronómetros en una *tabla* o *lista ligada* ordenada consiste en:

- Utilizar la *tabla de procesos* y cargar en ella un campo para su tiempo de expiración, si es que existe.

- Rastrear periódicamente la tabla de procesos para comparar el valor de cada cronómetro con el tiempo actual.

- Este tipo de algoritmos se denominan *algoritmos de barrido (sweep algorithms).*

8.15 Areas de Problemas en RPC

La RPC mediante el modelo C - S se utiliza ampliamente como base de los S. O. distribuidos [25, Tanenbaum].

Lo ideal es que la *RPC sea transparente:*

- El programador no debe poder decir si los *procedimientos* de biblioteca son *locales* o *remotos.*
- El programador debe poder escribir *procedimientos* sin importar si serán *ejecutados en forma local* o *remota.*
- La introducción de RPC en un sistema que se ejecutaba antes en una única cpu no debe ir acompañada de una serie de reglas que:
  - Prohiban construcciones antes válidas.
  - Exijan construcciones que antes eran opcionales.

La mayoría de los S. O. distribuidos no cumplen totalmente con estos criterios de ***transparencia****.*

Uno de los problemas es el de las *variables globales:*

- Ej.: en *UNIX* existe una variable global *errno:*
  - Luego de una llamada incorrecta al sistema, contiene un código que indica lo que estuvo mal.
  - Su existencia es información pública, ya que el estándar oficial de UNIX, *Posix*, exige que sea visible en uno de los archivos de encabezado imperativos, *errno.h.*
  - No se permite que una implantación lo oculte de los programadores.
  - Supongamos que un programador escribe dos procedimientos que tienen acceso directo a *errno:*
    - Uno se ejecuta en forma local, el otro en forma remota.
    - Uno de los procedimientos tendrá un acceso incorrecto.
  - El problema es que no se puede implantar el permiso para el acceso irrestricto de los procedimientos locales a las variables globales remotas y viceversa.
  - La *prohibición del acceso irrestricto* mencionado *viola el principio de transparencia:*
    - Los programas no deben actuar de manera distinta según RPC.

Otro problema son los *lenguajes* ***débilmente tipificados****,* como por ejemplo, *"C":*

- En un lenguaje fuertemente tipificado, como *"Pascal":*
  - El compilador y el procedimiento resguardo conocen todo lo relativo a todos los parámetros.

- *El resguardo puede ordenar los parámetros sin dificultad.*

- Con *"C"* se puede trabajar con arreglos sin especificar su tamaño:
  - *El resguardo del cliente no puede ordenar los parámetros:*
    - No puede determinar su tamaño.
  - Una solución consiste en que el programador defina el tamaño máximo cuando escriba la definición formal del servidor:
    - Resta flexibilidad y puede generar problemas de memoria.

Un problema adicional consiste en que *no siempre es posible **deducir los tipos de los parámetros**:*

- Ni siquiera a partir de una especificación formal del propio código, en especial considerando *"C".*
- La exclusión de *"C"* cuando se utilice RPC violaría la *transparencia.*

Conclusión:

- *El modelo C - S se ajusta a muchos casos pero no es perfecto.*

8.16 Comunicación en Grupo

Una hipótesis subyacente e intrínseca de RPC es que la comunicación solo es entre dos partes: el cliente y el servidor [25, Tanenbaum].

A veces existen circunstancias en las que *la comunicación es entre varios procesos* y no solo dos:[10]

- Ej.: un grupo de servidores de archivo que cooperan para ofrecer un único servicio de archivos tolerante a fallos:
  - Sería recomendable que un cliente envíe el mensaje a *todos* los servidores para garantizar la ejecución de la solicitud aunque alguno falle.
- *RPC no puede* controlar la comunicación de un servidor con muchos receptores, a menos que realice *RPC con cada uno* en forma individual.

Un grupo es una colección de procesos que actúan juntos en cierto sistema o alguna forma determinada por el usuario.

La *propiedad fundamental de todos los grupos* es que *cuando un mensaje se envía* al propio grupo, *todos los miembros del grupo lo reciben.*

Se trata de una *comunicación uno - muchos* (un emisor, muchos receptores), que se distingue de la *comunicación puntual* o *punto a punto* (un emisor, un receptor).

Los grupos son *dinámicos:*

[10] Ver Figura 8.11 de la página 281 [25, Tanenbaum].

- Se pueden crear y destruir.
- Un proceso se puede unir a un grupo o dejar a otro.
- Un proceso puede ser miembro de varios grupos a la vez.

La *implantación de la comunicación en grupo* depende en gran medida del *hardware:*

- En ciertas redes es posible crear una *dirección especial de red* a la que pueden escuchar *varias máquinas:*
  - Cuando se envía un mensaje a una de esas direcciones se lo entrega automáticamente a todas las máquinas que escuchan a esa dirección.
  - Esta técnica se denomina **multitransmisión.**
  - *Cada grupo* debe tener una *dirección de multitransmisión distinta.*

Las redes que no soportan multitransmisión operan con **transmisión simple:**

- Significa que los paquetes que tienen cierta dirección se entregan a todas las máquinas.
- Se puede utilizar para implantar los grupos, pero es *menos eficiente que la multitransmisión.*
- Cada máquina debe verificar, mediante su software, si el paquete va dirigido a ella:
  - En caso negativo se descarta, pero para analizarlo se generó una interrupción y se dedicó ciclos de cpu.

Otra solución es implantar la *comunicación en grupo mediante la transmisión por parte del emisor de paquetes individuales a cada uno de los miembros del grupo:*

- En vez de un paquete se precisan *"n"* paquetes.
- Es *menos eficiente* que las soluciones anteriores.
- Es una solución valida particularmente con *grupos pequeños.*
- *El envío de un mensaje de un emisor a un único receptor se llama* **unitransmisión.**

8.17 Aspectos del Diseño de la Comunicación en Grupo

En la comunicación en grupo también se presentan posibilidades tales como [25, Tanenbaum]:

- *Almacenamiento en buffers vs. el no almacenamiento.*
- *Bloqueo vs. no bloqueo.*

Además existen *otras posibilidades* que no se dan en la comunicación entre un emisor y un solo receptor.

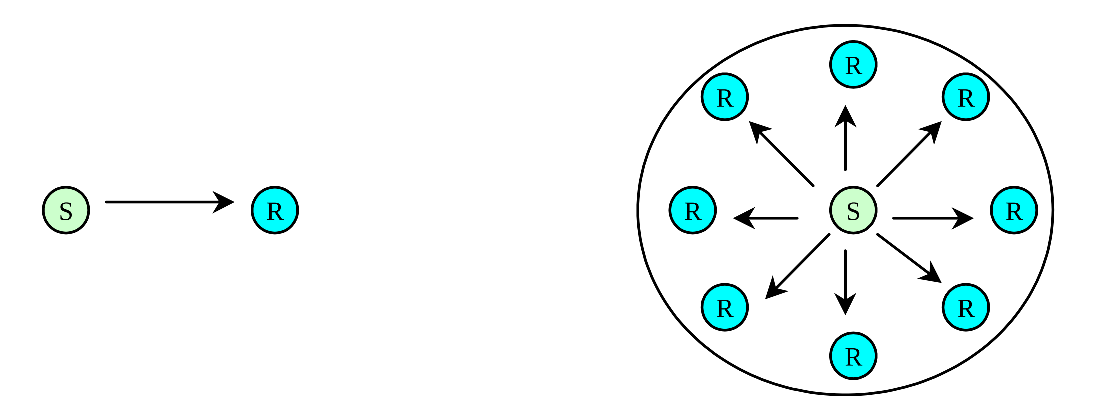


Figura 8.11: Comunicación punto a punto y comunicación uno a muchos.

8.17.1 Grupos Cerrados Vs. Grupos Abiertos

En los *grupos cerrados:*

- Solo los miembros del grupo pueden enviar hacia el grupo.
- Los extraños no pueden enviar mensajes al grupo como un todo, peor pueden enviar mensajes a miembros del grupo en lo individual.

En los *grupos abiertos:*

- Cualquier proceso del sistema puede enviar a cualquier grupo.

Los *grupos cerrados* se utilizan generalmente para el *procesamiento paralelo:*

- Ej.: un conjunto de procesos que trabajan de manera conjunta, tienen su propio objetivo y no interactúan con el mundo exterior.

Cuando la idea de grupo pretende soportar *servidores duplicados:*

- Es importante que los procesos que no sean miembros *(clientes)* puedan enviar hacia el grupo.
- Podría ser necesario que los miembros del grupo utilizaran la comunicación en grupo.

8.17.2 Grupos de Compañeros Vs. Grupos Jerárquicos

En algunos grupos todos *los procesos son iguales:*

- No existe distinción de jerarquías.
- Son los *grupos de compañeros.*

En otros grupos existe cierto tipo de *jerarquía:*

- Son los *grupos jerárquicos.*
- Ej.: un proceso es el *coordinador* y todos los demás son los *trabajadores.*
- Una solicitud de un trabajo que se genere en un cliente externo o en uno de los procesos trabajadores:
  - Se envía al coordinador.
  - El coordinador decide cuál de los trabajadores es el más adecuado para llevarla a cabo y se la envía.

Cada tipo de grupo tiene sus *ventajas y desventajas:*

- Respecto del *grupo de compañeros:*
  - Es simétrico y no tiene un punto de fallo.
  - Si uno de los procesos falla, el grupo se reduce pero puede continuar.
  - Para tomar decisiones grupales se producen retrasos debidos a la comunicación entre los miembros del grupo.
- Respecto del *grupo jerárquico:*
  - La pérdida del coordinador lleva al grupo a un alto total, lo que es una desventaja.
  - En condiciones normales, el coordinador funciona correctamente y toma decisiones sin molestar a los demás procesos.

Un ej. de *grupo jerárquico* podría ser un *programa de ajedrez en paralelo:*

- El *coordinador:*
  - Toma el tablero actual.
  - Genera todos los movimientos válidos a partir de él.
  - Los envía a los trabajadores para su evaluación.
  - Controla la estrategia de búsqueda.
  - Desarrolla el árbol del juego.
- Los *trabajadores:*
  - Al procesar las evaluaciones generan nuevos tableros.
  - Los tableros se envían al coordinador.
  - Al quedar inactivos, solicitan al coordinador un nuevo tablero en el cual trabajar.

8.17.3 Membresía del Grupo

La comunicación en grupo requiere cierto *método* para:

- Creación y eliminación de grupos.
- Unión y separación de procesos a grupos.

Una posibilidad es tener un *servidor de grupos* al cual enviar todas las solicitudes:

- Es un método directo, eficiente y fácil de implementar.
- La *desventaja* es el punto de fallo que representa la *administración centralizada* de los grupos.

Otra posibilidad es la *administración distribuida* de las membresías a grupos:

- En un grupo abierto, un proceso extraño puede enviar un mensaje a los integrantes del grupo para anunciar su presencia.
- En un grupo cerrado se precisa algo similar, ya que se debe contemplar la admisión de nuevos miembros al grupo cerrado.
- Al salir de un grupo, el proceso debe comunicarlo a los demás del grupo que deja.

Un *aspecto problemático* se presenta cuando *un miembro falla*, saliendo por lo tanto del grupo:

- No hay un anuncio apropiado de este hecho.
- Los demás miembros del grupo lo deben descubrir de forma experimental; luego se lo puede eliminar del grupo.

Otro aspecto importante es que la *entrada y salida al grupo debe sincronizarse* con el envío de mensajes:

- Un proceso que se unió a un grupo debe recibir todos los mensajes que se envíen al mismo.
- Un proceso que ha salido de un grupo:
  - No debe recibir más mensajes del grupo.
  - El grupo no debe recibir más mensajes del proceso.
  - Los otros miembros no deben recibir más mensajes del proceso saliente.
- Una forma de garantizar que una entrada o salida se integra al flujo de mensajes en el lugar correcto es convertir esta operación en un *mensaje a todo el grupo.*

Un *aspecto crítico* resulta cuando *fallan tantas máquinas que el grupo ya no puede funcionar:*

- Se necesita cierto protocolo para reconstruir el grupo.
- Alguno de los procesos deberá tomar la iniciativa.
- El protocolo deberá resolver la situación que se presenta cuando *dos o más procesos intentan al mismo tiempo reconstruir el grupo.*

8.17.4 Direccionamiento al Grupo

Los grupos deben poder direccionarse, al igual que los procesos.

Una forma es darle a *cada grupo una dirección única*, similar a una dirección de proceso.

Si la red soporta multitransmisión:

- La dirección del grupo se puede asociar con una *dirección de multitransmisión.*
- Cada mensaje enviado a la dirección del grupo se podrá multitransmitir.

Si la red no soporta multitransmisión:

- Se tendrá que utilizar transmisión simple.
- Cada núcleo lo recibirá y extraerá la dirección del grupo.
- Si ninguno de los procesos en la máquina es un miembro del grupo, se descarta la transmisión.
- En caso contrario se transfiere a todos los miembros del grupo.

Si la red no soporta multitransmisión ni transmisión simple:

- Se tendrá que utilizar *unitransmisión.*
- El núcleo de la máquina emisora deberá contar con una lista de las máquinas que tienen procesos pertenecientes al grupo.
- Deberá enviar a cada máquina un mensaje puntual.

Un *segundo método de direccionamiento de grupo* consiste en *pedirle al emisor una lista explícita de todos los destinos:*

- Ej.: lista de direcciones IP.
- El parámetro de la llamada *send* que especifica el destino es un *apuntador a una lista de direcciones.*
- La *desventaja* consiste en que los *procesos del usuario* (los miembros del grupo) deben tener conocimiento de *quién es miembro de cada grupo:*
  - *No es transparente.*
  - Los procesos del usuario deben actualizar las listas de miembros.

Un *tercer método* es el de *direccionamiento de predicados (predicate addressing):*

- El mensaje se envía a *todos los miembros del grupo* (o a todo el sistema) mediante uno de los métodos anteriores.
- El mensaje contiene un *predicado* (expresión booleana) para ser evaluado.
- El predicado puede utilizar el número de máquina del receptor, sus variables locales u otros factores.
- Si el valor del predicado es *"verdadero"* se acepta el mensaje y se descarta si es *"falso"*.
- Permite enviar un mensaje solo a aquellas máquinas que tengan al menos *"x"* mb de memoria libre y se puedan ocupar de un nuevo proceso.

8.17.5 Primitivas Send y Receive

El envío de un mensaje a un grupo ***no*** *se puede modelar como una llamada a un procedimiento.*

Con la comunicación en grupo existen en potencia *"n"* respuestas diferentes y no resulta aplicable el esquema de RPC.

Se utilizan llamadas explícitas para el envío y recepción (modelo de un solo sentido).

Si se *unifican las primitivas* puntuales y grupales para *send:*

- Uno de los parámetros indica el destino:
  - Si es una *dirección de un proceso*, se envía un único mensaje a ese proceso en particular.
  - Si es una *dirección de grupo* (o un apuntador a una lista de destinos), se envía un mensaje a todos los miembros del grupo.
- Un segundo parámetro apunta al mensaje por enviar.

Si se *fusionan las primitivas* puntuales y grupales para *receive:*

- La operación concluye cuando llega un mensaje puntual o un mensaje de grupo.

Si es necesario que las respuestas estén asociadas a solicitudes previas:

- Se envía un mensaje.
- Se efectúa varias veces un proceso *get_reply* para recolectar *todas las respuestas*, una a la vez.

8.17.6 Atomicidad

La mayoría de los *sistemas de comunicación en grupo* están diseñados para que *los mensajes enviados al grupo lleguen correctamente a todos los miembros o a ninguno de ellos:*

- Esta propiedad de *"todo o nada"* en la entrega se llama *atomicidad* o *transmisión atómica.*
- Facilita la programación de los sistemas distribuidos.
- Es de gran importancia para *garantizar la consistencia* de las bases de datos y de los archivos distribuidos y duplicados.

La única forma de garantizar que cada destino recibe todos sus mensajes es pedirle que envíe de regreso un reconocimiento después de recibir el mensaje:

- Esto funciona si las máquinas *no fallan.*
- *Si fallan:*
  - Algunos miembros del grupo habrán recibido el mensaje y otros no; esto es inaceptable.
  - Los miembros que no recibieron el mensaje ni siquiera saben que les falta algo, por lo que no pedirán una retransmisión; además, si pudieran detectar el faltante pero fallara el emisor, no podrán recibir el mensaje.

Una solución puede llegar del *algoritmo de Joseph y Birman:*

- Demuestra la *posibilidad de la transmisión atómica.*
- El emisor comienza con el envío de un mensaje a todos los miembros del grupo.
- Los cronómetros se activan y se envían las retransmisiones en los casos necesarios.
- Cuando un proceso recibe un mensaje:
  - Si no recibió ya este mensaje particular:
    - ∗ Lo envía a todos los miembros del grupo:
      - · Con cronómetros y retransmisiones en los casos necesarios.
  - Si ya recibió este mensaje particular:
    - ∗ No se efectúan envíos y el mensaje se descarta.

Este algoritmo asegura que todos los procesos sobrevivientes obtendrán el mensaje, independientemente del número de máquinas que fallen o el número de paquetes perdidos.

8.17.7 Ordenamiento de Mensajes

El ordenamiento de los mensajes es un *aspecto fundamental en la comunicación en grupo.*

Ej.: consideramos 5 máquinas, cada una con un proceso:

- Los procesos se identifican como 0, 1, 2, 3 y 4.
- Los procesos 0, 1, 3 y 4 pertenecen al mismo grupo.
- Los procesos 0 y 4 desean enviar un mensaje al grupo simultáneamente:
  - Supongamos que no se dispone de multitransmisión ni de transmisión simple.
  - Cada proceso debe enviar 3 mensajes independientes (unitransmisión).
- El proceso 0 envía a los procesos 1, 3 y 4.
- El proceso 4 envía a los procesos 0, 1 y 3.
- Una posible secuencia de intercalación de los mensajes es la siguiente:
  - 0 a 1; 4 a 0; 4 a 1; 4 a 3; 0 a 3; 0 a 4.

El problema es que cuando dos procesos contienden por el acceso a una LAN, el orden de envío de los mensajes ***no*** *es determinista.*

En el ejemplo anterior, los procesos 1 y 3 reciben los mensajes de los procesos 0 y 4 en distinto orden:

- Si los procesos 0 y 4 intentan actualizar el mismo registro de una base de datos, los procesos 1 y 3 terminarán con distintos valores finales.

Un sistema debe tener una semántica bien definida con respecto al orden de entrega de los mensajes.

La mejor garantía es la *entrega inmediata de todos los mensajes*, en el *orden en que fueron enviados:*

- Todos los mensajes destinados a un grupo deben entregarse antes de comenzar a entregar la siguiente serie de mensajes.
- Todos los receptores reciben todos los mensajes en el mismo orden.
- Es esquema se denomina *ordenamiento con respecto al tiempo global.*

Una *variante* al esquema anterior es el *ordenamiento consistente:*

- Si dos mensajes *"a"* y *"b"* se envían muy cercanos en el tiempo, el sistema:
  - Elige uno de ellos como el "primero".
  - Lo envía a todos los miembros del grupo.
  - Luego envía el otro mensaje.
- Se garantiza que los mensajes lleguen a todos los miembros del grupo *en el mismo orden:*
  - Dicho orden podría no ser aquel con el que fueron enviados.

8.17.8 Grupos Traslapados

Un proceso puede ser miembro de varios grupos a la vez, pero esto puede generar un *nuevo tipo de inconsistencia.*

Ej.: supongamos que:

- El grupo 1 está formado por los procesos *"A", "B"* y *"C".*
- El grupo 2 está formado por los procesos *"B", "C"* y *"D".*
- Cada uno de los procesos *"A"* y *"D"* decide de manera simultánea enviar un mensaje a sus grupos respectivos.
- El sistema utiliza el ordenamiento con respecto al tiempo global dentro de cada grupo.
- Se utiliza la unitransmisión.
- El orden de los mensajes es el siguiente:
  - *"A"* a *"B"; "D"* a *"B"; "D"* a *"C" y "A"* a *"C".*
  - Se da la situación de que dos procesos, *"B"* y *"C"*, reciben los mensajes en un *orden distinto.*

El *problema* es el siguiente:

- Existe un ordenamiento con respecto al tiempo global dentro de cada grupo.
- *No existe coordinación* entre varios grupos.
- Resulta muy complicado implantar el orden con respecto al tiempo entre los distintos grupos, pero no siempre es necesario hacerlo.

8.17.9 Escalabilidad

Es necesario *considerar cómo funcionarán los algoritmos* cuando se tienen los siguientes casos:

- Grupos con centenas o miles de miembros.
- Centenas o miles de grupos.
- Utilización de varias *LAN* y *compuertas (gateways)* conectadas, es decir *interred (internetwork).*
- Diseminación de grupos en varios continentes.

Las compuertas pueden afectar la *multitransmisión* y el hecho requerido por algunas redes de tener un solo paquete en la red en un instante dado.

Capítulo 9

Sincronización en Sistemas Distribuidos

9.1 Introducción a la Sincronización en Sistemas Distribuidos

Además de la comunicación, *es fundamental la forma* en que los procesos [25, Tanenbaum]:

- *Cooperan.*
- Se *sincronizan* entre sí.

Ejemplos:

- Forma de implantar las *regiones críticas.*
- Forma de *asignar recursos* en un sistema distribuido.

Los problemas relativos a las *regiones críticas, exclusión mutua* y *la sincronización:*

- Generalmente se resuelven en sistemas de una sola cpu con métodos como los *semáforos* y los *monitores:*
  - Se basan en la *memoria compartida.*
  - No son aplicables a *sistemas distribuidos.*

Otro *problema de gran importancia* es el *tiempo y la forma de medirlo*, ya que juega un papel *fundamental* en algunos modelos de sincronización.

9.2 Sincronización de Relojes

Generalmente los *algoritmos distribuidos* tienen las siguientes *propiedades [25, Tanenbaum]:*

- La información relevante se distribuye entre varias máquinas.

- Los procesos toman las decisiones solo con base en la información disponible en forma local.
- Debe evitarse un único punto de fallo en el sistema.
- No existe un *reloj común* o alguna otra *fuente precisa del tiempo global.*

Los primeros tres puntos indican que es inaceptable reunir toda la información en un solo lugar para su procesamiento, pero *lograr la sincronización sin centralización* requiere hacer las cosas *distintas* al caso de los sistemas operativos tradicionales.

El último punto también es crucial:

- En un sistema centralizado el tiempo no es ambiguo.
- En un sistema distribuido no es trivial poner de acuerdo a todas las máquinas en la hora.
- Se requiere un *acuerdo global en el tiempo*, pues la falta de sincronización en los relojes puede ser drástica en procesos dependientes del tiempo.

La pregunta es si es posible sincronizar ***todos*** *los relojes en un sistema distribuido.*

9.3 Relojes Lógicos

Las computadoras poseen un circuito para el registro del tiempo conocido como *dispositivo reloj [25, Tanenbaum].*

Es un *cronómetro* consistente en un cristal de cuarzo de precisión sometido a una tensión eléctrica que:

- Oscila con una frecuencia bien definida que depende de:
  - Al forma en que se corte el cristal.
  - El tipo de cristal.
  - La magnitud de la tensión.
- A cada cristal se le asocian dos registros:
  - Registro *contador.*
  - Registro *mantenedor.*
- Cada oscilación del cristal decrementa en *"1"* al contador.
- Cuando el contador llega a *"0":*
  - Se genera una *interrupción.*
  - El contador se vuelve a cargar mediante el registro mantenedor.

- Se puede *programar un cronómetro* para que genere *una interrupción "x" veces por segundo.*

- Cada *interrupción* se denomina *marca de reloj.*

Para una computadora y un reloj:

- No interesan pequeños desfasajes del reloj porque:

  - Todos los procesos de la máquina usan el mismo reloj y tendrán *consistencia interna.*
  - Importan los *tiempos relativos.*

Para varias computadoras con sus respectivos relojes:

- Es imposible garantizar que los cristales de computadoras distintas oscilen con la misma frecuencia.

- Habrá una *pérdida de sincronía en los relojes (de software)*, es decir que tendrán *valores distintos* al ser leidos.

La *diferencia entre los valores del tiempo* se llama *distorsión del reloj* y podría generar fallas en los programas dependientes del tiempo.

Lamport demostró que la sincronización de relojes es posible y presentó un algoritmo para lograrlo.

Lamport señaló que *la sincronización de relojes no tiene que ser absoluta:*

- Si 2 procesos no interactúan no es necesario que sus relojes estén sincronizados.

- Generalmente lo importante no es que los procesos estén de acuerdo en la hora, pero sí importa que coincidan en el *orden en que ocurren los eventos.*

Para ciertos algoritmos lo que importa es la *consistencia interna de los relojes:*

- No interesa su cercanía particular al tiempo real (oficial).

- Los relojes se denominan *relojes lógicos.*

Los *relojes físicos* son relojes que:

- Deben ser iguales (estar sincronizados).

- No deben desviarse del *tiempo real* más allá de cierta magnitud.

Para *sincronizar los relojes lógicos*, **Lamport** definió la relación *ocurre antes de (happens-before):*

- Si *"a"* y *"b"* son eventos en el mismo proceso y *"a"* ocurre antes de *"b"*, entonces *"a –> b"* es *verdadero.*

- "Ocurre antes de" es una *relación transitiva:*
  - Si *"a -> b"* y *"b -> c"*, entonces *"a -> c".*
- Si dos eventos *"x"* e *"y"* están en *procesos diferentes que no intercambian mensajes*, entonces *"x -> y" no es verdadero*, pero *tampoco lo es "y -> x":*
  - Se dice que son *eventos concurrentes.*

Necesitamos una forma de medir el tiempo tal que a cada evento *"a"*, le podamos asociar un valor del tiempo *"C(a)"* en el que *todos los procesos* estén de acuerdo:

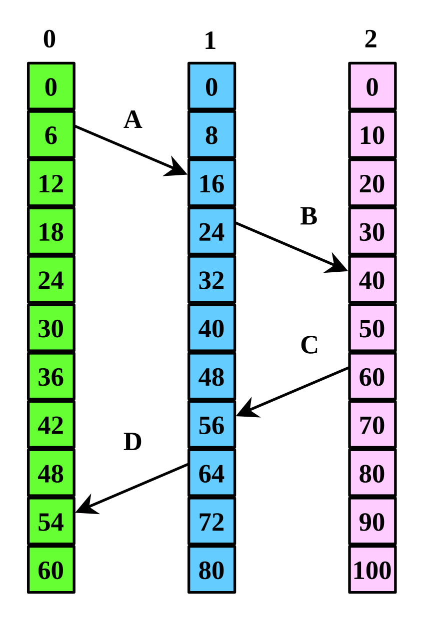
- Se debe cumplir que:
  - Si *"a -> b"* entonces *"C(a) < C(b)".*
  - El *tiempo del reloj*, *"C"*, siempre debe ir *hacia adelante* (creciente), y nunca hacia atrás (decreciente).

El *algoritmo de Lamport* asigna tiempos a los eventos.

Consideramos *tres procesos* que se ejecutan en *diferentes máquinas*, cada una con su propio reloj y velocidad:[1]

- El proceso *"0"* envía el mensaje *"a"* al proceso *"1"* cuando el reloj de *"0"* marca *"6".*
- El proceso *"1"* recibe el mensaje *"a"* cuando su reloj marca *"16".*
- Si el mensaje acarrea el tiempo de inicio *"6"*, el proceso *"1"* considerará que tardó 10 marcas de reloj en viajar.
- El mensaje *"b"* de *"1"* a *"2"* tarda 16 marcas de reloj.
- El mensaje *"c"* de *"2"* a *"1"* sale en *"60"* y llega en *"56"*, tardaría un *tiempo negativo*, lo cual es *imposible.*
- El mensaje *"d"* de *"1"* a *"0"* sale en *"64"* y llega en *"54".*
- Lamport utiliza la relación *"ocurre antes de":*
  - Si *"c"* sale en *"60"* debe llegar en *"61"* o en un tiempo posterior.
  - Cada mensaje acarrea el tiempo de envío, de acuerdo con el reloj del emisor.
  - Cuando un mensaje llega y el reloj del receptor muestra un *valor anterior* al tiempo en que se envió el mensaje:
    * El receptor *adelanta su reloj* para que tenga una unidad más que el tiempo de envío.

[1] Ver Figura 9.1 de la página 293 [25, Tanenbaum].



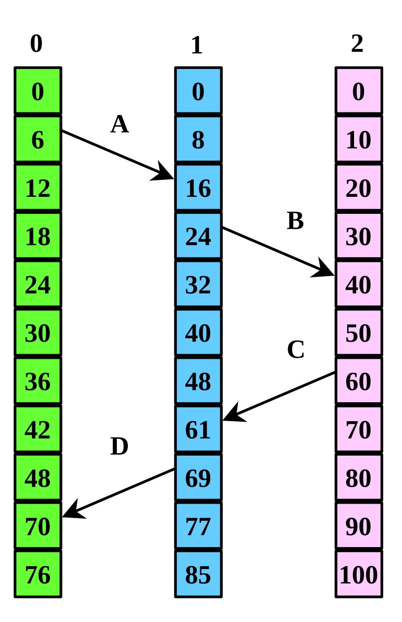


Figura 9.1: Ejemplo de tres procesos cuyos relojes corren a diferentes velocidades - El algoritmo de Lamport corrige los relojes.

Este algoritmo cumple nuestras necesidades para el *tiempo global*, si se hace el siguiente *agregado:*

- Entre dos eventos cualesquiera, el reloj debe marcar al menos una vez.
- Dos eventos no deben ocurrir *exactamente* al mismo tiempo.

Con este algoritmo *se puede asignar un tiempo a todos los eventos en un sistema distribuido*, con las siguientes *condiciones:*

- Si *"a"* ocurre antes de *"b"* en el mismo proceso, *"C(a) < C(b)".*
- Si *"a"* y *"b"* son el envío y recepción de un mensaje, *"C(a) < C(b)".*
- Para todos los eventos *"a"* y *"b"*, *"C(a)"* es distinto de *"C(b)".*

9.4 Relojes Físicos

El algoritmo de Lamport proporciona un orden de eventos sin ambigüedades, pero [25, Tanenbaum]:

- Los valores de tiempo asignados a los eventos no tienen porqué ser cercanos a los tiempos reales en los que ocurren.
- En ciertos sistemas (ej.: *sistemas de tiempo real*), es importante la *hora real* del reloj:

- Se precisan *relojes físicos externos* (más de uno).
- Se deben *sincronizar:*
    * Con los relojes del mundo real.
    * Entre sí.

La medición del tiempo real con alta precisión no es sencilla.
Desde antiguo el tiempo se ha medido *astronómicamente.*

Se considera el *día solar* al intervalo entre dos tránsitos consecutivos del sol, donde el tránsito del sol es el evento en que el sol alcanza su punto aparentemente más alto en el cielo.

El *segundo solar* se define como 1 / 86.400 *de un día solar.*

Como el *período de rotación de la tierra no es constante*, se considera el *segundo solar promedio* de un gran número de días.

Los físicos definieron al *segundo* como el tiempo que tarda el átomo de cesio 133 para hacer 9.192.631.770 *transiciones:*

- Se tomó este número para que el *segundo atómico* coincida con el *segundo solar promedio de 1958.*

La **Oficina Internacional de la Hora en París** (**BIH**) recibe las indicaciones de cerca de 50 relojes atómicos en el mundo y calcula el *tiempo atómico internacional* (**TAI**).

Como consecuencia de que el *día solar promedio* (**DSP**) es cada vez mayor, un *día TAI* es 3 mseg menor que un *DSP:*

- La BIH introduce *segundos de salto* para hacer las correcciones necesarias para que permanezcan *en fase:*
    - El sistema de tiempo basado en los segundos TAI.
    - El movimiento aparente del sol.
- Surge el *tiempo coordenado universal* (**UTC**).

El **Instituto Nacional del Tiempo Estándar** (**NIST**) de EE. UU. y de otros países:

- Operan estaciones de radio de onda corta o satélites de comunicaciones.
- Transmiten pulsos UTC con cierta regularidad establecida (cada segundo, cada 0,5 mseg, etc.).
- Se deben conocer con precisión la posición relativa del emisor y del receptor:
    - Se debe compensar el retraso de propagación de la señal.
    - Si la señal se recibe por módem también se debe compensar por la ruta de la señal y la velocidad del módem.
    - Se dificulta la obtención del tiempo con una precisión extremadamente alta.

9.5 Algoritmos Para la Sincronización de Relojes

Si *una máquina* tiene un *receptor de UTC*, todas las máquinas deben sincronizarse con ella [25, Tanenbaum].

Si *ninguna máquina* tiene un *receptor de UTC:*

- Cada máquina lleva el registro de su propio tiempo.
- Se debe mantener el tiempo de todas las máquinas *tan cercano como sea posible.*

Se supone que cada máquina tiene un *cronómetro* que provoca una interrupción *"h"* veces por segundo.

Cuando el cronómetro se detiene, el manejador de interrupciones añade *"1"* a un *reloj en software.*

El *reloj en software* mantiene un registro del *número de marcas (interrupciones)* a partir de cierta fecha acordada antes; al valor de este reloj se lo llama *"C".*

Cuando el *tiempo UTC* es *"t"*, el valor del reloj en la máquina *"p"* es *"Cp(t)":*

- *Lo ideal* sería que *"Cp(t)"* = *"t"* para toda *"p"* y todo *"t":*
  - *"dC/dt"* debería ser *"1".*
- *Lo real* es que los cronómetros no realizan interrupciones *exactamente* *"h"* veces por segundo:
  - Poseen un *error relativo* de aproximadamente 10^{-5}.
  - El fabricante especifica una constante *"ρ"* llamada *tasa máxima de alejamiento* que *acota el error.*
  - El cronómetro funciona dentro de su especificación si existe una constante *"ρ"* y se cumple:
    - $1 - \rho \leqq dC\ /\ dt \leqq 1 + \rho$.

Si dos relojes se alejan de UTC en la dirección opuesta:

- En un instante Δt luego de la sincronización podrían estar tan alejados como: 2 ρ Δt.
- Para garantizar que no difieran más de δ:
  - Se deben volver a sincronizar (en software) al menos cada δ / 2 ρ segundos.

9.5.1 Algoritmo de Cristian

Es adecuado para sistemas en los que:

- Una máquina tiene *un receptor UTC*, por lo que se la llama *despachador del tiempo.*
- El objetivo es *sincronizar todas* las máquinas con ella.

Cada máquina envía un mensaje al servidor para solicitar el tiempo actual, periódicamente, en un tiempo no mayor que δ / 2 ρ segundos.

El *despachador del tiempo* responde prontamente con un mensaje que contiene el *tiempo actual "C_{UTC}".*

Cuando el emisor obtiene la respuesta puede hacer que su tiempo sea *"C_{UTC}".*

Un gran problema es que *el tiempo no puede correr hacia atrás:*

- *"C_{UTC}"* no puede ser menor que el tiempo actual *"C"* del emisor.
- La atención del requerimiento en el *servidor de tiempos* requiere un tiempo del manejador de interrupciones.
- También se debe considerar el tiempo de transmisión.

El cambio del reloj se debe introducir de manera *global:*

- Si el cronómetro genera 100 interrupciones por segundo:
  - Cada interrupción añade 10 mseg al tiempo.
  - Para atrasar solo agregaría 9 mseg.
  - Para adelantar agregaría 11 mseg.

La corrección por el tiempo del servidor y el tiempo de transmisión se hace midiendo *en el emisor:*

- El tiempo inicial (envío) *"T_0".*
- El tiempo final (recepción) *"T_1".*
- Ambos tiempos se miden con *el mismo reloj.*

El *tiempo de propagación del mensaje* será *(T_1 - T_0) / 2.*

Si el *tiempo del servidor* para manejar la interrupción y procesar el mensaje es *"I":*

- El *tiempo de propagación* será *(T_1 - T_0 - I) / 2.*

Para *mejorar la precisión:*

- Se toman varias mediciones.
- Se descartan los valores extremos.
- Se promedia el resto.

El *tiempo de propagación* se suma al *tiempo del servidor* para *sincronizar al emisor* cuando éste recibe la respuesta.

9.5.2 Algoritmo de Berkeley

En el *algoritmo de Cristian* el servidor de tiempo es *pasivo.*

En el *algoritmo de Berkeley* el servidor de tiempo:

- Es *activo.*
- Realiza un muestreo periódico de *todas* las máquinas para preguntarles el tiempo.
- Con las respuestas:
  - Calcula un tiempo promedio.
  - Indica a las demás máquinas que avancen su reloj o disminuyan la velocidad del mismo hasta lograr la disminución requerida.

Es adecuado cuando *no se dispone de un receptor UTC*.

9.5.3 Algoritmos con Promedio

Los anteriores son *algoritmos centralizados.*

Una clase de *algoritmos descentralizados* divide el tiempo en *intervalos de resincronización* de longitud fija:

- El i-ésimo intervalo:
  - Inicia en *"T_0 + i R"* y va hasta *"T_0 + (i + 1) R".*
  - *"T_0"* es un momento ya acordado en el pasado.
  - *"R"* es un parámetro del sistema.

Al inicio de cada intervalo cada máquina transmite el *tiempo actual* según su reloj.

Debido a la diferente velocidad de los relojes *las transmisiones no serán simultáneas.*

Luego de que una máquina transmite su hora, inicializa un cronómetro local para reunir las demás transmisiones que lleguen en cierto intervalo *"S".*

Cuando recibe todas las transmisiones se ejecuta un algoritmo para calcular una *nueva hora* para los relojes.

Una variante es *promediar los valores* de todas las demás máquinas.

Otra variante es *descartar los valores extremos antes de promediar* (los *"m"* mayores y los *"m"* menores).

Una mejora al algoritmo considera la *corrección por tiempos de propagación.*

9.5.4 Varias Fuentes Externas de Tiempo

Los sistemas que requieren una *sincronización muy precisa con UTC* se pueden equipar con *varios receptores de UTC.*

Las distintas fuentes de tiempo generaran *distintos rangos (intervalos de tiempo)* donde *"caerán"* los respectivos UTC, por lo que *es necesaria una sincronización.*

Como la transmisión no es instantánea se genera una cierta *incertidumbre en el tiempo.*

Cuando un procesador obtiene todos los rangos de UTC:

- Verifica si alguno de ellos es ajeno a los demás y de serlo lo descarta por ser un valor extremo.
- Calcula la intersección (en el tiempo) de los demás rangos.
- La intersección determina un intervalo cuyo punto medio será el UTC y la hora del reloj interno.

Se deben *compensar* los retrasos de transmisión y las diferencias de velocidades de los relojes.

Se debe *asegurar* que el tiempo no corra hacia atrás.

Se debe *resincronizar* periódicamente desde las fuentes externas de UTC.

9.6 Exclusión Mutua

Cuando un proceso debe leer o actualizar ciertas *estructuras de datos compartidas [25, Tanenbaum]:*

- Primero ingresa a una *región crítica* para lograr la *exclusión mutua* y garantizar que ningún otro proceso utilizará las estructuras de datos al mismo tiempo.

En sistemas monoprocesadores las regiones críticas se protegen con semáforos, monitores y similares.

En sistemas distribuidos la cuestión es más compleja.

9.6.1 Un Algoritmo Centralizado

La forma más directa de lograr la *exclusión mutua en un sistema distribuido* es simular a la forma en que se lleva a cabo en un sistema monoprocesador.

Se elige un *proceso coordinador.*

Cuando un proceso desea *ingresar a una región crítica:*

- Envía un mensaje de solicitud al coordinador:
  - Indicando la región crítica.
  - Solicitando permiso de acceso.
- Si ningún otro proceso está en ese momento en esa región crítica:
  - El coordinador envía una respuesta otorgando el permiso.
- Cuando llega la respuesta el proceso solicitante entra a la región crítica.

Si un proceso pide permiso para *entrar a una región crítica ya asignada a otro proceso:*

- El coordinador *no* otorga el permiso y encola el pedido.

Cuando un proceso *sale de la región crítica* envía un mensaje al coordinador para liberar su acceso exclusivo:

- El coordinador extrae el primer elemento de la cola de solicitudes diferidas y envía a ese proceso un mensaje otorgando el permiso, con lo cual el proceso queda habilitado para acceder a la región crítica solicitada.

Es un esquema sencillo, justo y con pocos mensajes de control.

La limitante es que *el coordinador puede ser un cuello de botella* y puede fallar y bloquear a los procesos que esperan una respuesta de habilitación de acceso.

9.6.2 Un Algoritmo Distribuido

El objetivo es no tener un único punto de fallo (el coordinador central).

Un ej. es el *algoritmo de Lamport* mejorado por *Ricart y Agrawala.*

Se requiere un *orden total de todos los eventos* en el sistema para saber cuál ocurrió primero.

Cuando un proceso desea *entrar a una región crítica:*

- Construye un mensaje con el nombre de la región crítica, su número de proceso y la hora actual.
- Envía el mensaje a todos los demás procesos y de manera conceptual a él mismo.
- Se supone que cada mensaje tiene un reconocimiento.

Si el receptor *no está en la región crítica y no desea entrar a ella*, envía de regreso un mensaje o.k. al emisor.

Si el receptor *ya está en la región crítica* no responde y encola la solicitud.

Si el receptor *desea entrar a la región crítica pero aún no lo logró*, compara:

- La marca de tiempo del mensaje recibido con,
- La marca contenida en el mensaje que envió a cada uno.
- La menor de las marcas gana.
- Si el mensaje recibido es menor el receptor envía un o.k.
- Si su propio mensaje tiene una marca menor el receptor no envía nada y encola el pedido.

Luego de enviar las solicitudes un proceso:

- Espera hasta que alguien más obtiene el permiso.
- Cuando llegan todos los permisos puede entrar a la región crítica.

Cuando *un proceso sale de la región crítica:*

- Envía mensajes o.k. a todos los procesos en su cola.
- Elimina a todos los elementos de la cola.

La exclusión mutua queda garantizada sin bloqueo ni inanición.

El número de mensajes necesarios por entrada es *"2(n - 1)"*, siendo *"n"* el número total de procesos en el sistema.

No existe un único punto de fallo sino "n":

- Si cualquier proceso falla no responderá a las solicitudes.
- La falta de respuesta se interpretará como negación de acceso:
  - Se *bloquearán* los siguientes intentos de los demás procesos por entrar a todas las regiones críticas.

Se *incrementa la probabilidad de fallo en "n" veces* y también el *tráfico en la red.*

Se puede *solucionar el bloqueo* si:

- El emisor espera y sigue intentando hasta que regresa una respuesta o,
- El emisor concluye que el destinatario está fuera de servicio.

Otro *problema* es que:

- Se utilizará una primitiva de comunicación en grupo o,
- Cada proceso debe mantener la lista de miembros del grupo, incluyendo los procesos que ingresan, los que salen y los que fallan.
- Se complica para gran número de procesos.

Un importante *problema adicional* es que:

- *Todos* los procesos participan en *todas* las decisiones referentes a las entradas en las regiones críticas.
- Se *sobrecarga* el sistema.

Una *mejora* consiste en permitir que un proceso entre a una región crítica *con el permiso de una mayoría simple de los demás procesos* (en vez de todos):

- Luego de que un proceso otorgó el permiso a otro para entrar a una región crítica, no puede otorgar el mismo permiso a otro proceso hasta que el primero libere su permiso.

9.6.3 Un Algoritmo de Anillo de Fichas (Token Ring)

Los procesos se organizan por software formando un anillo lógico asignándose a cada proceso una posición en el anillo.

Cada proceso sabe cuál es el siguiente luego de él.

Al *inicializar el anillo* se le da al proceso *"0"* una **ficha (token)** que circula en todo el anillo, que se transfiere del proceso *"k"* al *"k + 1"* en mensajes puntuales.

Cuando un proceso obtiene la ficha de su vecino *verifica si intenta entrar a una región crítica:*

- En caso *positivo:*
  - El proceso entra a la región crítica, hace el proceso necesario y sale de ella.
  - Después de salir pasa la ficha a lo largo del anillo:
    * No se puede entrar a una segunda región crítica con la misma ficha (token o permiso).
- En caso *negativo:*
  - La vuelve a pasar.

En un instante dado solo un proceso puede estar en una región crítica.
Si la ficha se pierde debe ser regenerada, pero es difícil detectar su perdida:

- La cantidad de tiempo entre las apariciones sucesivas de la ficha en la red no está acotada, por ello es difícil decidir si está perdida o demorada en algún proceso que no la libera.

La *falla de un proceso* es detectada cuando su vecino intenta sin éxito pasarle la ficha:

- Se lo debe eliminar del grupo y pasar la ficha al siguiente proceso activo.
- Todos los procesos deben mantener la *configuración actual* del anillo.

9.7 Algoritmos de Elección

Son los algoritmos para la elección de un proceso coordinador, iniciador, secuenciador, etc. [25, Tanenbaum].

El *objetivo* de un algoritmo de elección es garantizar que iniciada una elección ésta concluya con el *acuerdo de todos los procesos* con respecto a la identidad del *nuevo coordinador.*

9.7.1 El Algoritmo del Grandulón o de García-Molina

Un *proceso "P"* inicia una elección cuando observa que *el coordinador ya no responde* a las solicitudes.

"P" realiza una elección de la siguiente manera:

- Envía un mensaje *elección* a los demás procesos con un número mayor.
- Si nadie responde asume que gana la elección y se convierte en el *nuevo coordinador.*
- Si un proceso con un número mayor responde, *toma el control* y el trabajo de *"P"* termina.

Un proceso puede recibir en cualquier momento un mensaje *elección* de otros procesos con un número menor:

- Envía de regreso un mensaje o.k. al emisor para indicar que está vivo y que tomará el control.
- Realiza una elección salvo que ya esté haciendo alguna.

En cierto momento *todos* los procesos han declinado ante uno de ellos, que será el *nuevo coordinador*, que envía un mensaje *coordinador* a todos los procesos para anunciarlo.

Si un proceso inactivo se activa realiza una elección:

- Si él tiene el número más alto será el *nuevo coordinador:*
  - Siempre gana el proceso que posee el número mayor, de ahí el nombre *"algoritmo del grandulón".*

9.7.2 Un Algoritmo de Anillo

Se supone que los procesos tienen un orden físico o lógico, es decir que cada proceso conoce a su sucesor.

Cuando algún proceso observa que *el coordinador no funciona:*

- Construye un mensaje *elección* con su propio número de proceso.
- Envía el mensaje a su sucesor.
- Si el sucesor está inactivo:
  - El emisor va hacia el siguiente número del anillo o al siguiente de éste.
  - Continúa hasta localizar un proceso en ejecución.
  - En cada paso, al emisor añade su propio número de proceso a la lista en el mensaje.
- En cierto momento el mensaje regresa al proceso que lo inició:
  - El proceso lo reconoce al recibir un mensaje con su propio número de proceso.
- El mensaje de *elección* se transforma en mensaje *coordinador* y circula nuevamente:
  - Informa a los demás procesos:
    * Quién es el *coordinador*, es decir, el miembro de la lista con el número mayor.
    * Quiénes son los *miembros del nuevo anillo.*
- Concluida la ronda de información el *mensaje coordinador* se elimina y continúan los procesos.

9.8 Transacciones Atómicas

Las *técnicas de sincronización* ya vistas son de *bajo nivel* [25, Tanenbaum]:

- El *programador* debe enfrentarse directamente con los detalles de:
  - La exclusión mutua.
  - El manejo de las regiones críticas.
  - La prevención de bloqueos.
  - La recuperación de fallas.

Se precisan *técnicas de abstracción de mayor nivel* que:

- Oculten estos aspectos técnicos.
- Permitan a los programadores *concentrarse en los algoritmos* y la forma en que los procesos trabajan juntos en paralelo.

Tal abstracción la llamaremos *transacción atómica, transacción* o *acción atómica.*
La principal propiedad de la transacción atómica es el "todo o nada":

- O se hace todo lo que se tenía que hacer como una unidad o no se hace nada.
- Ejemplo:
  - Un cliente llama al Banco mediante una PC con un módem para:
    * Retirar dinero de una cuenta.
    * Depositar el dinero en otra cuenta.
  - La operación tiene dos etapas.
  - Si la conexión telefónica falla luego de la primer etapa pero antes de la segunda:
    * Habrá un retiro pero no un depósito.
  - La solución consiste en agrupar las dos operaciones en una *transacción atómica:*
    * Las dos operaciones terminarían o no terminaría ninguna.
    * *Se debe regresar al estado inicial si la transacción no puede concluir.*

9.9 El Modelo de Transacción

Supondremos que [25, Tanenbaum]:

- El sistema consta de *varios procesos independientes* que pueden fallar aleatoriamente.
- El software subyacente maneja *transparentemente* los errores de comunicación.

9.9.1 Almacenamiento Estable

Se puede implantar con *una pareja de discos comunes.*

Cada bloque de la unidad 2 es una copia exacta (espejo) del bloque correspondiente en la unidad 1.

Cuando se actualiza un bloque:

- Primero se actualiza y verifica el bloque de la unidad 1.
- Luego se actualiza y verifica el bloque de la unidad 2.

Si el sistema falla luego de actualizar la unidad 1 y antes de actualizar la unidad 2:

- Luego de la recuperación se pueden comparar ambos discos bloque por bloque:
  - Se puede actualizar la unidad 2 en función de la 1.

Si se detecta el deterioro espontáneo de un bloque, se lo regenera partiendo del bloque correspondiente en la otra unidad.

Un esquema de este tipo es adecuado para las aplicaciones que requieren de un *alto grado de tolerancia de fallos*, por ej. las *transacciones atómicas.*

9.9.2 Primitivas de Transacción

Deben ser *proporcionadas por el sistema operativo o por el sistema de tiempo de ejecución del lenguaje.*

Ejemplos:

- *Begin_ transaction:* los comandos siguientes forman una transacción.
- *End_ transaction:* termina la transacción y se intenta un compromiso.
- *Abort_ transaction:* se elimina la transacción; se recuperan los valores anteriores.
- *Read:* se leen datos de un archivo (o algún otro objeto).
- *Write:* se escriben datos en un archivo (o algún otro objeto).

Las operaciones entre *Begin* y *End* forman el *cuerpo de la transacción* y deben ejecutarse *todas o ninguna* de ellas:

- Pueden ser llamadas al sistema, procedimiento de biblioteca o enunciados en un lenguaje.

9.9.3 Propiedades de las Transacciones

Las *propiedades fundamentales* son:

- *Serialización:*
  - Las transacciones concurrentes no interfieren entre sí.
- *Atomicidad:*
  - Para el mundo exterior, la transacción ocurre de manera indivisible.
- *Permanencia:*
  - Una vez *comprometida* una transacción, los cambios son permanentes.

La **serialización** garantiza que si dos o más transacciones se ejecutan al mismo tiempo:

- El resultado final aparece como si *todas l*as transacciones se ejecutasen de manera secuencial *en cierto orden:*
  - Para cada una de ellas y para los demás procesos.

La **atomicidad** garantiza que cada transacción no ocurre o bien se realiza en su totalidad:

- Se presenta como una *acción indivisible e instantánea.*

La **permanencia** se refiere a que una vez comprometida una transacción:

- Sigue adelante y los resultados son *permanentes.*

9.9.4 Transacciones Anidadas

Se presentan cuando *las transacciones pueden contener subtransacciones* (procesos hijos) que:

- Se ejecuten en paralelo entre sí en procesadores distintos.
- Pueden originar nuevas subtransacciones.

9.10 Implantación del Modelo de Transacción

Existen varios métodos de implantación [25, Tanenbaum].

9.10.1 Espacio de Trabajo Particular

Consiste en que cuando un *proceso inicia una transacción* se le otorga un *espacio de trabajo particular:*

- Contiene todos los archivos (y otros objetos) a los cuales tiene acceso.
- Las lecturas y escrituras irán a este espacio hasta que la transacción se *complete o aborte:*
  - El *"espacio real"* es el *sistema de archivos normal.*
- Significa alto consumo de recursos por las copias de los objetos al espacio de trabajo particular.

Cuando un proceso *inicia* una transacción, basta crear un espacio de trabajo particular para él que sea vacío excepto por un apuntador de regreso al espacio de trabajo de su *proceso padre.*

Para una *transacción del nivel superior* el espacio de trabajo del *padre* es el *sistema de archivos "real".*

Cuando el proceso *abre un archivo para lectura*, se siguen los apuntadores de regreso hasta localizar el archivo en el espacio de trabajo del padre (o algún antecesor).

Cuando se *abre un archivo para escritura:*

- Se lo localiza de manera similar que para lectura.
- Se copia en primer lugar al espacio de trabajo particular:
  - Una optimización consiste en copiar solo el índice del archivo en el espacio de trabajo particular:
    * El índice es el bloque de datos asociado a cada archivo e indica la localización de sus bloques en el disco.
    * Es el nodo-i correspondiente.

La *lectura* por medio del índice particular (del espacio de trabajo particular) no es problemática, pues las direcciones en disco a las que referencia son las originales.

La *modificación* de un bloque de un archivo requiere:

- Hacer una copia del bloque.
- Insertar en el índice la dirección de la copia.

La modificación sobre la copia no afecta al bloque original.

Un tratamiento similar se da al *agregado* de bloques; los nuevos bloques se llaman *bloques sombra (shadow blocks).*

El proceso que ejecuta la transacción ve el archivo modificado pero los demás procesos ven el archivo original.

Si la transacción aborta (termina anormalmente):

- El espacio de trabajo particular se elimina.
- Los bloques particulares a los que apunta se colocan nuevamente en la lista de bloques libres.

Si la transacción se compromete (termina normalmente):

- Los índices particulares se desplazan al espacio de trabajo del padre de manera atómica.
- Los bloques que no son alcanzables se colocan en la lista de bloques libres.

9.10.2 Bitácora de Escritura Anticipada

Este método también se denomina *lista de intenciones.*

Los archivos *realmente se modifican* pero antes de cambiar cualquier bloque:

- Se graba un registro en la *bitácora ("log")* de escritura anticipada en un *espacio de almacenamiento estable:*
  - Se indica la transacción que produce el cambio, el archivo y bloque modificados y los valores anterior y nuevo.

Si la transacción tiene éxito y se hace un compromiso:

- Se escribe un registro del compromiso en la bitácora.
- Las estructuras de datos no tienen que modificarse, puesto que ya han sido actualizadas.

Si la transacción aborta:

- Se puede utilizar la bitácora para respaldo del estado original:
  - A partir del final y hacia atrás:
    - Se lee cada registro de la bitácora.
    - Se deshace cada cambio descripto en él.
- Esta acción se denomina *retroalimentación.*

Por medio de la bitácora se puede:

- Ir hacia adelante (realizar la transacción).
- Ir hacia atrás (deshacer la transacción).

9.10.3 Protocolo de Compromiso de Dos Fases (Two - Phase Commit)

Uno de los procesos que intervienen funciona como el *coordinador.*

El coordinador escribe una entrada en la *bitácora* para indicar que *inicia el protocolo.*

El coordinador envía a cada uno de los procesos relacionados (subordinados) un mensaje para que estén preparados para el compromiso.

Cuando un *subordinado* recibe el mensaje:

- Verifica si está listo para comprometerse.
- Escribe una entrada en la bitácora.
- Envía de regreso su decisión.

Cuando el coordinador ha recibido *todas las respuestas* sabe si debe establecer el compromiso o abortar:

- Si todos los procesos están listos para comprometerse cierra la transacción.
- Si alguno de los procesos no se compromete (o no responde) la transacción se aborta.
- El coordinador:
  - Escribe una entrada en la bitácora.
  - Envía un mensaje a cada subordinado para informarle de la decisión.

9.11 Control de Concurrencia en el Modelo de Transacción

Los algoritmos de control de concurrencia son necesarios cuando se ejecutan varias transacciones de manera simultánea [25, Tanenbaum]:

- En distintos procesos.
- En distintos procesadores.

Los *principales algoritmos* son:

- *El de la cerradura.*
- *El del control optimista de la concurrencia.*
- *El de las marcas de tiempo.*

9.11.1 Cerradura (locking)

Cuando un *proceso* debe *leer o escribir* en un archivo (u otro objeto) como parte de una transacción, primero *cierra el archivo.*

La *cerradura* se puede hacer mediante:

- Un único manejador *centralizado* de cerraduras.
- Un manejador *local* de cerraduras en cada máquina.

El *manejador de cerraduras:*

- Mantiene una lista de los archivos cerrados.
- Rechaza todos los intentos por cerrar archivos ya cerrados por otros procesos.

El sistema de transacciones generalmente adquiere y libera las cerraduras *sin acción* por parte del programador.

Una *mejora* consiste en distinguir las *cerraduras para lectura* de las *cerraduras para escritura.*

Una *cerradura para lectura* no impide otras cerraduras para lectura:

- Las cerraduras para lectura *se comparten.*

Una *cerradura para escritura* sí impide otras cerraduras (de lectura o de escritura):

- Las cerraduras para escritura *no se comparten*, es decir que deben ser exclusivas.

El *elemento por cerrar* puede ser un archivo, un registro, un campo, etc. y lo relativo al *tamaño del elemento* por cerrar se llama la *granularidad de la cerradura.*

Mientras más fina sea la granularidad:

- Puede ser más precisa la cerradura.
- Se puede lograr un mayor paralelismo en el acceso al recurso.
- Se requiere un mayor número de cerraduras.

Generalmente se utiliza la *cerradura de dos fases:*

- El proceso adquiere todas las cerraduras necesarias durante la *fase de crecimiento.*
- El proceso las libera en la *fase de reducción.*

Se deben evitar situaciones de *aborto en cascada:*

- Se graba en un archivo y luego se libera su cerradura.
- Otra transacción lo cierra, realiza su trabajo y luego establece un compromiso.
- La transacción original aborta.
- La segunda transacción (ya comprometida) debe deshacerse, ya que sus resultados se basan en un archivo que no debería haber visto cuando lo hizo.

Las *cerraduras* comunes y de dos fases *pueden provocar bloqueos* cuando dos procesos intentan adquirir la misma pareja de cerraduras pero en orden opuesto, por lo tanto se deben utilizar técnicas de prevención y de detección de bloqueos para superar el problema.

9.11.2 Control Optimista de la Concurrencia

La idea es muy sencilla:

- Se sigue adelante y se hace todo lo que se deba hacer, sin prestar atención a lo que hacen los demás.
- Se actúa a posteriori si se presenta algún problema.

Se mantiene un *registro de los archivos leídos o grabados.*
En el momento del *compromiso:*

- Se verifican todas las demás transacciones para ver si alguno de los archivos ha sido modificado desde el inicio de la transacción:
  - Si esto ocurre la transacción aborta.
  - Si esto no ocurre se realiza el compromiso.

Las *principales ventajas* son:

- Ausencia de bloqueos.
- Paralelismo máximo ya que no se esperan cerraduras.

La *principal desventaja* es:

- Re-ejecución de la transacción en caso de falla.
- La probabilidad de fallo puede crecer si la carga de trabajo es muy alta.

9.11.3 Marcas de Tiempo

Se asocia a cada transacción una *marca de tiempo* al iniciar *(begin_ transaction).*

Se garantiza que las marcas son únicas mediante el *algoritmo de Lamport.*

Cada archivo del sistema tiene asociadas una *marca de tiempo para la lectura* y otra para la *escritura*, que indican la última transacción *comprometida* que realizó la lectura o escritura.

Cuando un *proceso* intente acceder a un *archivo*, lo logrará si las marcas de tiempo de lectura y escritura son menores (más antiguas) que la marca de la transacción activa.

Si la marca de tiempo de la transacción activa es menor que la del archivo que intenta acceder:

- Una transacción iniciada posteriormente ha accedido al archivo y ha efectuado un compromiso.
- La transacción activa se ha realizado tarde y se aborta.

En el método de las marcas no preocupa que las transacciones concurrentes utilicen los mismos archivos, pero sí importa que la transacción con el número menor esté en primer lugar.

Las marcas de tiempo tienen *propiedades distintas a las de los bloqueos:*

- Una transacción aborta cuando encuentra una marca mayor (posterior).
- En iguales circunstancias y en un esquema de cerraduras podría esperar o continuar inmediatamente.

Las marcas de tiempo son libres de bloqueos, lo que es una gran ventaja.

9.11.4 Resumen

Los diferentes esquemas ofrecen distintas ventajas pero el problema principal es la gran complejidad de su implantación.

9.12 Bloqueos en Sistemas Distribuidos

Son peores que los bloqueos en sistemas monoprocesador [25, Tanenbaum]:

- Son más difíciles de evitar, prevenir, detectar y solucionar.
- Toda la información relevante está dispersa en muchas máquinas.

Son *especialmente críticos* en sistemas de *bases de datos distribuidos.*
Las *estrategias usuales* para el manejo de los bloqueos son:

- *Algoritmo del avestruz:*
  - Ignorar el problema.
- *Detección:*
  - Permitir que ocurran los bloqueos, detectarlos e intentar recuperarse de ellos.
- *Prevención:*
  - Hacer que los bloqueos sean imposibles desde el punto de vista estructural.
- *Evitarlos:*
  - Evitar los bloqueos mediante la asignación cuidadosa de los recursos.

El *algoritmo del avestruz* merece las mismas consideraciones que en el caso de mono-procesador.

En los sistemas distribuidos resulta muy difícil implantar algoritmos para evitar los bloqueos:

- Se requiere saber de antemano la proporción de cada recurso que necesitará cada proceso.
- Es muy difícil disponer de esta información en forma práctica.

Las técnicas más aplicables para el *análisis de los bloqueos* en sistemas distribuidos son:

- Detección.
- Prevención.

9.13 Detección Distribuida de Bloqueos

Cuando se detecta un bloqueo en un S. O. convencional se resuelve eliminando uno o más procesos [25, Tanenbaum].

Cuando se detecta un *bloqueo en un sistema basado en transacciones atómicas* se resuelve *abortando una o más transacciones:*

- El sistema restaura el estado que tenía antes de iniciar la transacción.
- La transacción puede volver a comenzar.

Las consecuencias de la *eliminación de un proceso* son mucho menos severas si se utilizan las *transacciones* que en caso de que no se utilicen.

9.13.1 Detección Centralizada de Bloqueos

Cada máquina mantiene la *gráfica de recursos* de sus propios procesos y recursos.

Un *coordinador central* mantiene la *gráfica de recursos de todo el sistema*, que es la unión de todas las gráficas individuales.

Cuando el coordinador detecta un ciclo *elimina uno de los procesos* para romper el bloqueo.

La *información de control* se debe transmitir explícitamente, existiendo las siguientes variantes:

- Cada máquina informa cada actualización al coordinador.
- Cada máquina informa periódicamente las modificaciones desde la última actualización.
- El coordinador requiere la información cuando la necesita.

La información de control *incompleta o retrasada* puede llevar a *falsos bloqueos:*

- El coordinador interpreta erróneamente que existe un bloqueo y elimina un proceso.
- Una posible solución es utilizar el *algoritmo de Lamport* para disponer de un *tiempo global.*

9.13.2 Detección Distribuida de Bloqueos

Un algoritmo típico es el de Chandy-Misra-Haas.

Los procesos pueden solicitar varios recursos (por ejemplo cerraduras) *al mismo tiempo*, en vez de uno cada vez.

Se permiten las *solicitudes simultáneas* de varios procesos:

- Un proceso puede esperar a *uno o más recursos* simultáneamente.
- Los recursos que espera un proceso pueden ser *locales o remotos* (de otra máquina).

Si el proceso *"0"* se bloquea debido al proceso *"1":*

- Se genera un *mensaje de exploración* que se envía al proceso (o procesos) que detienen los recursos necesarios.
- El mensaje consta de tres números:
  - El proceso recién bloqueado, el proceso que envía el mensaje y el proceso al cual se envía.
- Al llegar el mensaje el receptor verifica si él mismo espera a algunos procesos, en cuyo caso:
  - El mensaje se actualiza:
    * Se conserva el primer campo.
    * Se reemplaza el segundo por su propio número de proceso y el tercero por el número del proceso al cual espera.
  - El mensaje se envía al proceso debido al cual se bloquea:
    * Si se bloquea debido a varios procesos les envía mensajes (diferentes) a todos ellos.
- Si un mensaje recorre todo el camino y regresa a su emisor original (el proceso enlistado en el primer campo), entonces:
  - Existe un ciclo y el sistema está bloqueado.

Una *forma de romper el bloqueo* es que el *proceso que inició la exploración se comprometa a suicidarse* y, si varios procesos se bloquean al mismo tiempo e inician exploraciones, todos ellos se suicidarán.

Una variante es eliminar solo al proceso del ciclo que tiene el número más alto.

9.14 Prevención Distribuida de Bloqueos

La prevención consiste en el diseño cuidadoso del sistema para que los bloqueos sean imposibles estructuralmente [25, Tanenbaum].

Entre las distintas técnicas se incluye:

- Permitir a los procesos que solo conserven un recurso a la vez.
- Exigir a los procesos que soliciten todos sus recursos desde un principio.
- Hacer que todos los procesos liberen todos sus recursos cuando soliciten uno nuevo.

En un *sistema distribuido con tiempo global y transacciones atómicas:*

- Se puede asociar a cada *transacción* una *marca de tiempo global* al momento de su inicio.
- No pueden haber parejas de transacciones con igual marca de tiempo asociada.

La idea es que cuando un proceso está a punto de bloquearse en espera de un recurso que está utilizando otro proceso:

- Se verifica cuál de ellos tiene la marca de tiempo mayor (es más joven).
- Se puede permitir la espera solo si el proceso en estado de espera tiene una marca inferior (más viejo) que el otro.

Al seguir cualquier *cadena de procesos en espera:*

- Las marcas aparecen en forma creciente.
- Los ciclos son imposibles.

Otra posibilidad es permitir la espera de procesos *solo si el proceso que espera tiene una marca mayor (es más joven)* que el otro proceso; las marcas aparecen en la cadena en forma descendente.

Es más sabio *dar prioridad a los procesos más viejos:*

- Se ha invertido tiempo de proceso en ellos.
- Probablemente conservan más recursos.

Capítulo 10

Procesos y Procesadores en Sistemas Distribuidos

10.1 Introducción a los Hilos (Threads)

Muchos S. O. distribuidos soportan *múltiples hilos de control dentro de un proceso* que [25, Tanenbaum]:

- Comparten un *único espacio de direcciones.*
- Se ejecutan *quasi - paralelamente* como si fueran procesos independientes.

Ej.: *servidor de archivos* que debe bloquearse ocasionalmente en espera de acceso al disco:

- Si tiene varios hilos de control podría ejecutar un segundo hilo mientras el primero espera:
  - El resultado sería mejor rendimiento y desempeño.
  - No se logra esto con procesos servidores independientes puesto que deben compartir un buffer caché común y deben estar en el mismo espacio de direcciones.

En muchos sentidos *los hilos son como miniprocesos:*

- Cada hilo:
  - Se ejecuta en forma estrictamente secuencial.
  - Tiene su propio contador de programa y una pila para llevar un registro de su posición.
- Los hilos comparten la cpu de la misma forma que lo hacen los procesos:
  - Secuencialmente, en *tiempo compartido.*
- Solo en un *multiprocesador* se pueden ejecutar *realmente en paralelo.*

- Los hilos pueden crear hilos hijos.
- Mientras un hilo está bloqueado se puede ejecutar otro hilo del mismo proceso.

Los distintos hilos de un proceso *comparten* un espacio de direcciones, el conjunto de archivos abiertos, los procesos hijos, cronómetros, señales, etc.

Los hilos pueden tener *distintos estados:* en ejecución, bloqueado, listo, terminado.

10.2 Uso de Hilos

Los hilos permiten la combinación del paralelismo con la ejecución secuencial y el bloqueo de las llamadas al sistema [25, Tanenbaum].

Consideramos el ejemplo del *servidor de archivos* con sus posibles organizaciones para muchos hilos de ejecución.

Iniciamos con el *modelo servidor / trabajador:*

- Un hilo, el *servidor*, lee las solicitudes de trabajo en el buzón del sistema.
- Elige a un hilo *trabajador* inactivo (bloqueado) y le envía la solicitud, despertándolo.
- El hilo trabajador verifica si puede satisfacer la solicitud por medio del bloque caché compartido, al que tienen acceso todos los hilos.
- Si no envía un mensaje al disco para obtener el bloque necesario y se duerme esperando el fin de la operación.
- Se llama:
  - Al planificador y se inicializa otro hilo, que tal vez sea el servidor, para pedir más trabajo; o.
  - A otro trabajador listo para realizar un trabajo.

Los hilos ganan un desempeño considerable pero cada uno de ellos se programa en forma secuencial.

Otro *modelo* es el *de equipo:*

- Todos los *hilos son iguales* y cada uno obtiene y procesa sus propias solicitudes.
- *No hay servidor.*
- Se utiliza una cola de trabajo que contiene todos los trabajos pendientes, que son trabajos que los hilos no han podido manejar.
- Un hilo debe verificar primero la cola de trabajo antes de buscar en el buzón del sistema.

Un tercer *modelo* es el *de entubamiento:*

- El primer hilo genera ciertos datos y los transfiere al siguiente para su procesamiento.

- Los datos pasan de hilo en hilo y en cada etapa se lleva a cabo cierto procesamiento.

Un programa diseñado adecuadamente y que utilice hilos debe funcionar *bien:*

- En una *única cpu* con hilos compartidos.
- En un verdadero *multiprocesador.*

10.3 Aspectos del Diseño de un Paquete de Hilos

Un conjunto de primitivas relacionadas con los hilos (ej.: llamadas a biblioteca) disponibles para los usuarios se llama un "paquete de hilos" [25, Tanenbaum].

Respecto del *manejo de los hilos* se tienen *hilos estáticos* e *hilos dinámicos.*

En un **diseño estático:**

- Se elige el número de hilos al escribir el programa o durante su compilación.
- Cada uno de ellos tiene asociada una pila fija.
- Se logra simplicidad pero también inflexibilidad.

En un **diseño dinámico:**

- Se permite la creación y destrucción de los hilos durante la ejecución.
- La llamada para la creación de hilos determina:
  - El programa principal del hilo.
  - Un tamaño de pila.
  - Una prioridad de planificación, etc.
- La llamada generalmente regresa un identificador de hilo:
  - Se usará en las posteriores llamadas relacionadas al hilo.
- Un proceso:
  - Se inicia con un solo hilo.
  - Puede crear el número necesario de hilos.

Los hilos pueden concluir:

- Por su cuenta, al terminar su trabajo.
- Por su eliminación desde el exterior.

Los hilos *comparten una memoria común:*

- Contiene datos que los distintos hilos comparten.
- El acceso generalmente se controla mediante regiones críticas.

10.4 Implantación de un Paquete de Hilos

Un paquete de hilos se puede implantar *en el espacio [25, Tanenbaum]:*

- *Del usuario.*
- *Del núcleo.*

Implantación del paquete de hilos en el espacio del usuario:

- El núcleo no sabe de su existencia.
- El núcleo maneja procesos con un único hilo.
- No requiere soporte de hilos por parte del S. O.
- Los hilos se ejecutan en un sistema de *tiempo de ejecución:*
  - Es un grupo de procedimientos que manejan los hilos.
- Cuando un hilo ejecuta una llamada al sistema o cualquier acción que pueda provocar su suspensión:
  - Llama a un procedimiento del sistema de tiempo de ejecución.
  - El procedimiento verifica si hay que suspender al hilo, en cuyo caso:
    * Almacena los registros del hilo en una tabla.
    * Busca un hilo no bloqueado para ejecutarlo.
    * Vuelve a cargar los registros de la máquina con los valores resguardados del nuevo hilo.
- Las *principales ventajas* son:
  - El intercambio de hilos es más rápido que si se utilizaran los señalamientos al núcleo.
  - Cada proceso puede tener su propio algoritmo adaptado de planificación de hilos.
  - Tienen una mejor escalabilidad para un número muy grande de hilos, ya que no afectan al núcleo con tablas y bloques de control (pila).

Implantación del paquete de hilos en el espacio del núcleo:

- No se necesita un sistema de tiempo de ejecución.
- Para cada proceso el núcleo tiene una tabla con una entrada por cada hilo que contiene:
  - Los registros, estados, prioridades y demás información relativa al hilo.

- Todas las llamadas que pueden bloquear un hilo se implantan como llamadas al sistema:
  - Significa un costo mayor (en recursos y tiempo).
- Cuando un hilo se bloquea, el núcleo puede ejecutar:
  - Otro hilo listo del mismo proceso.
  - Un hilo de otro proceso:
    * Con los hilos a nivel usuario el sistema de tiempo de ejecución mantiene en ejecución los hilos de su propio proceso hasta que:
      · El núcleo les retira la cpu, o.
      · No hay hilos listos.

Un *problema fundamental de los paquetes de hilos a nivel usuario* es el de las *llamadas al sistema con bloqueo:*

- No se puede permitir que el hilo realmente realice la llamada al sistema:
  - Detendría a todos los hilos del proceso.
  - Un hilo bloqueado no debe afectar a los demás.
- Una solución es agregar código junto a la llamada al sistema para verificar si la misma no generaría bloqueo:
  - Se efectuaría la llamada al sistema solo si la verificación da o.k.
  - El código adicional suele llamarse *jacket.*

Otro *problema de los paquetes de hilos a nivel usuario* es que *si un hilo comienza su ejecución no puede ejecutarse ningún otro hilo de ese proceso*, salvo que el hilo entregue voluntariamente la cpu.

Un problema adicional para los *hilos a nivel usuario* es que *generalmente los programadores desean los hilos en aplicaciones donde los hilos se bloquean a menudo:*

- Ej.: servidor de archivos con varios hilos.

10.5 Hilos y RPC

Es común que los sistemas distribuidos utilicen RPC e hilos [25, Tanenbaum].

Al iniciar un hilo servidor, *"S"*, éste *exporta su interfaz* al informarle de ésta al núcleo; la interfaz define los procedimientos que puede llamar, sus parámetros, etc.

Al iniciar un hilo cliente, *"C"*, éste *importa la interfaz* del núcleo:

- Se le proporciona un identificador especial para utilizarlo en la llamada.
- El núcleo sabe que *"C"* llamará posteriormente a *"S":*

- Crea estructuras de datos especiales para prepararse para la llamada.

Una de las estructuras es una *pila de argumentos compartida* por *“C”* y *“S”*, que se asocia de manera lectura / escritura en ambos espacios de direcciones.

Para *llamar al servidor, “C”:*

- Coloca sus argumentos en la pila compartida mediante el procedimiento normal de transferencia.
- Hace un señalamiento al núcleo colocando un identificador especial en un registro.

El *núcleo:*

- Detecta esto y deduce que es una llamada local.
- Modifica el mapa de memoria del cliente para colocar éste en el espacio de direcciones del servidor.
- Inicia el hilo cliente, al ejecutar el procedimiento del servidor.

La *llamada* se efectúa de tal forma que:

- Los argumentos se encuentran ya en su lugar:
  - No es necesario su copiado u ordenamiento.
  - La RPC local se puede realizar más rápido de esta manera.

10.6 Modelos de Sistemas

En un sistema distribuido, con ***varios procesadores****, un aspecto fundamental del diseño es* ***cómo*** *se los utiliza [25, Tanenbaum].*

Los procesadores distribuidos se pueden organizar de varias *formas:*

- *Modelo de estación de trabajo.*
- *Modelo de la pila de procesadores.*
- *Modelo híbrido.*

10.7 El Modelo de Estación de Trabajo

El sistema consta de *estaciones de trabajo (PC) dispersas conectadas entre sí mediante una red de área local (LAN) [25, Tanenbaum].*

Pueden contar o no con disco rígido en cada una de ellas.

Los *usuarios* tienen:

- Una cantidad fija de poder de cómputo exclusiva.
- Un alto grado de autonomía para asignar los recursos de su estación de trabajo.

Uso de los *discos* en las estaciones de trabajo:

- *Sin disco:*
  - Bajo costo, fácil mantenimiento del hardware y del software, simetría y flexibilidad.
  - Gran uso de la red, los servidores de archivos se pueden convertir en cuellos de botella.
- *Disco para paginación y archivos de tipo borrador:*
  - Reduce la carga de la red respecto del caso anterior.
  - Alto costo debido al gran número de discos necesarios.
- *Disco para paginación, archivos de tipo borrador y archivos binarios (ejecutables):*
  - Reduce aún más la carga sobre la red.
  - Alto costo y complejidad adicional para actualizar los binarios.
- *Disco para paginación, borrador, binarios y ocultamiento de archivos:*
  - Reduce aún más la carga de red y de los servidores de archivos.
  - Alto costo.
  - Problemas de consistencia del caché.
- Sistema local de archivos completo:
  - Escasa carga en la red.
  - Elimina la necesidad de los servidores de archivos.
  - Pérdida de transparencia.

10.8 Uso de Estaciones de Trabajo Inactivas

La idea consiste en ordenar remotamente la ejecución de procesos en estaciones de trabajo inactivas [25, Tanenbaum].

Los *aspectos clave* son:

- ¿Cómo encontrar una estación de trabajo inactiva?.
- ¿Cómo lograr que un proceso remoto se ejecute de forma *transparente*?.
- ¿Qué ocurre si regresa el poseedor de la máquina?.

Generalmente se considera que una estación de trabajo está *"inactiva"* cuando se dan ambas condiciones:

- Nadie toca el ratón o el teclado durante varios minutos.
- No se ejecuta algún proceso iniciado por el usuario.

Los *algoritmos para localizar las estaciones de trabajo inactivas* se pueden dividir en dos categorías:

- *Controlados por el servidor.*
- *Controlados por el cliente.*

Algoritmos controlados por el servidor:

- Cuando una estación de trabajo está inactiva:
  - Se convierte en un *servidor potencial.*
  - Anuncia su disponibilidad:
    - Proporciona su nombre, dirección en la red y propiedades:
      - Grabándolos en un archivo, o.
      - Transmitiéndolos a las otras estaciones.
- Se pueden dar situaciones de *competencia entre distintos usuarios* para acceder a la misma estación inactiva al mismo tiempo:
  - Se deben detectar al ingresar el requerimiento.
  - Solo progresa el primer requerimiento arribado.
  - Se elimina a la estación de la lista de inactivas.
  - Quien hizo el llamado puede enviar su ambiente e iniciar el proceso remoto.

Algoritmos controlados por el cliente:

- El cliente transmite una solicitud indicando el programa que desea ejecutar, la cantidad de memoria necesaria, si requiere un chip coprocesador, etc.
- Al regresar la respuesta se elige una estación y se la configura.

Para *ejecutar el proceso en la estación remota* seleccionada se debe lograr:

- El desplazamiento del código.
- La configuración del proceso remoto de modo que:
  - "Vea" el mismo ambiente que tendría en el caso local, en la *estación de trabajo de origen.*
  - Ejecute de la misma forma que en el caso local.

Se necesita *la misma visión del sistema de archivos*, el mismo directorio de trabajo, etc.

Si se trabaja sobre el *servidor de archivos* se envían las solicitudes de disco al servidor.

Si se trabaja con *discos locales* se envían las solicitudes a la máquina de origen para su ejecución.

Ciertas operaciones como la lectura del teclado y la escritura en la pantalla:

- Nunca se pueden ejecutar en la máquina remota.
- Deben regresar a la máquina de origen.

Todas las *llamadas al sistema* que soliciten el estado de la máquina deben realizarse en la máquina *donde se ejecuta el proceso.*

Las *llamadas al sistema relacionadas con el tiempo* son un serio problema debido a las dificultades de *sincronización.*

En caso de que *regrese el poseedor de la máquina:*

- Se podría no hacer nada, contra la idea de estaciones de trabajo *"personales".*
- Se podría eliminar el proceso intruso:
  - Abruptamente, perdiéndose el trabajo hecho y generando caos en el sistema de archivos.
  - Ordenadamente, salvando el procesamiento ya hecho y preservando la integridad del sistema de archivos.
- Se podría emigrar el proceso a otra estación.

10.9 El Modelo de la Pila de Procesadores

Se dispone de un *conjunto de cpu que se pueden asignar dinámicamente* a los usuarios según la demanda [25, Tanenbaum].

Los usuarios *no disponen de estaciones de trabajo* sino de *terminales gráficas de alto rendimiento.*

No existe el concepto de *propiedad* de los procesadores, los que pertenecen a todos y se utilizan *compartidamente.*

El principal *argumento para la centralización del poder de cómputo* como una pila de procesadores proviene de la *teoría de colas:*

- Llamamos *"λ"* a la *tasa de entradas totales de solicitudes por segundo* de todos los usuarios combinados.
- Llamamos *"μ"* a la *tasa de procesamiento de solicitudes* por parte del servidor.
- Para una **operación estable** debe darse que *"$\mu > \lambda$":*
  - Se pueden permitir *pequeños lapsos de tiempo* en los que la tasa de entrada exceda a la de servicio.

- Llamamos *"T"* al *promedio de tiempo entre la emisión de una solicitud y la obtención de una respuesta completa:*
  - $T = 1 / (\mu - \lambda)$.
  - Cuando *"λ"* tiende a *"0"*, *"T"* no tiende a *"0"*.
- Supongamos que tenemos *"n"* multiprocesadores personales, cada uno con cierto número de cpu y con su propio sistema de colas con tasas *"λ"* y *"μ"* y tiempo *"T":*
  - Si reunimos todas las cpu y formamos *una sola pila de procesadores* tendremos *un solo sistema de colas* en vez de *"n"* colas ejecutándose en paralelo.
  - La *tasa de entrada* será *"n λ"*, la *tasa de servicio* será *"n μ"* y el *tiempo promedio de respuesta* será:
    * $T_1 = 1 / (n\ \mu - n\ \lambda) = 1 / n\ (\mu - \lambda) = T / n$.
  - **Conclusión:** si reemplazamos *"n"* pequeños recursos por uno grande que sea *"n"* veces más poderoso:
    * *Podemos reducir el tiempo promedio de respuesta "n" veces.*

El modelo de pila es más eficiente que el modelo de búsqueda de estaciones inactivas.

También existe el **modelo híbrido** que consta de *estaciones de trabajo* y *una pila de procesadores.*

10.10 Asignación de Procesadores

Son necesarios algoritmos para decidir *cuál proceso hay que ejecutar y en qué máquina [25, Tanenbaum].*

Para el modelo de estaciones de trabajo:

- Decidir cuándo ejecutar el proceso de manera local y cuándo buscar una estación inactiva.

Para el modelo de la pila de procesadores:

- Decidir dónde ejecutar cada nuevo proceso.

10.11 Modelos de Asignación

Generalmente se utilizan las siguientes *hipótesis* [25, Tanenbaum]:

- Todas las máquinas son idénticas (o al menos compatibles en el código); difieren a lo sumo en la velocidad.
- Cada procesador se puede comunicar con los demás.

Las *estrategias de asignación de procesadores* se dividen en:

- **No migratorias:**
  - Una vez colocado un proceso en una máquina permanece ahí hasta que termina.
- **Migratorias:**
  - Un proceso se puede trasladar aunque haya iniciado su ejecución.
  - Permiten un mejor balance de la carga pero son más complejas.

Los *algoritmos de asignación* intentan *optimizar algo:*

- **Uso de las cpu:**
  - Maximizar el número de ciclos de cpu que se ejecutan para trabajos de los usuarios.
  - Minimizar el tiempo de inactividad de las cpu.
- **Tiempo promedio de respuesta:**
  - Minimizar no los tiempos individuales de respuesta sino los tiempos promedio de respuesta.
- **Tasa de respuesta:**
  - Minimizar la tasa de respuesta, que es el tiempo necesario para ejecutar un proceso en cierta máquina dividido por el tiempo que tardaría en cierto procesador de referencia.

10.12 Aspectos del Diseño de Algoritmos de Asignación de Procesadores

Los *principales aspectos* son los siguientes [25, Tanenbaum]:

- *Algoritmos deterministas vs. heurísticos.*
- *Algoritmos centralizados vs. distribuidos.*
- *Algoritmos óptimos vs. subóptimos.*
- *Algoritmos locales vs. globales.*
- *Algoritmos iniciados por el emisor vs. iniciados por el receptor.*

Los *algoritmos deterministas* son adecuados cuando se sabe anticipadamente *todo* acerca del comportamiento de los procesos, pero esto generalmente no se da, aunque puede haber en ciertos casos aproximaciones estadísticas.

Los *algoritmos heurísticos* son adecuados cuando la carga es impredecible.

Los *diseños centralizados* permiten reunir toda la información en un lugar y tomar una mejor decisión; la desventaja es que la máquina central se puede sobrecargar y se pierde robustez ante su posible falla.

Generalmente los *algoritmos óptimos* consumen más recursos que los *subóptimos*, además, en la mayoría de los sistemas reales se buscan soluciones subóptimas, heurísticas y distribuidas.

Cuando se va a crear un proceso se debe decidir si se ejecutará *en la máquina que lo genera o en otra (política de transferencia):*

- La decisión se puede tomar *"solo con información local"* o *"con información global".*
- Los *algoritmos locales* son sencillos pero no óptimos.
- Los *algoritmos globales* son mejores pero consumen muchos recursos.

Cuando una máquina se deshace de un proceso la *política de localización* debe decidir dónde enviarlo:

- Necesita información de la carga en todas partes, obteniéndola de:
  - Un emisor sobrecargado que busca una máquina inactiva.
  - Un receptor desocupado que busca trabajo.

10.13 Aspectos de la Implantación de Algoritmos de Asignación de Procesadores

Casi todos los algoritmos suponen que *las máquinas conocen su propia carga y que pueden informar su estado [25, Tanenbaum]:*

- La *medición de la carga* no es tan sencilla.
- Un método consiste en contar el número de procesos (hay que considerar los procesos latentes no activos).
- Otro método consiste en contar solo los procesos en ejecución o listos.
- También se puede medir la fracción de tiempo que la cpu está ocupada.

Otro aspecto importante es el *costo excesivo en consumo de recursos para recolectar medidas y desplazar procesos,* ya que se debería considerar el tiempo de cpu, el uso de memoria y el ancho de banda de la red utilizada por el algoritmo para asignación de procesadores.

Se debe considerar la *complejidad del software* en cuestión y sus implicancias para el desempeño, la correctez y la robustez del sistema.

Si el uso de un algoritmo sencillo proporciona casi la misma ganancia que uno más caro y más complejo, generalmente será mejor utilizar *el más sencillo.*

Se debe otorgar gran importancia a la *estabilidad del sistema:*

- Las máquinas ejecutan sus algoritmos en forma asíncrona por lo que el sistema nunca se equilibra.

- La mayoría de los algoritmos que intercambian información:

  - Son correctos luego de intercambiar la información y de que todo se ha registrado.
  - Son poco confiables mientras las tablas continúan su actualización, es decir que se presentan situaciones de no equilibrio.

10.14 Ejemplos de Algoritmos de Asignación de Procesadores

10.14.1 Un Algoritmo Determinista Según la Teoría de Gráficas

Es aplicable a sistemas donde *se conoce* [25, Tanenbaum]:

- Requerimientos de cpu y de memoria de los procesos.

- Tráfico promedio entre cada par de procesos.

Si el número de procesos supera al número de cpu:

- Habrá que asignar varios procesos a la misma cpu.

- La asignación deberá *minimizar el tráfico en la red.*

El sistema se puede representar en una *gráfica con pesos:*

- Cada nodo es un proceso.

- Cada arco es el flujo de mensajes entre dos procesos.

El problema es encontrar la forma de *partir la gráfica en subgráficas sujetas a restricciones* (ej.: de cpu y de memoria):[1]

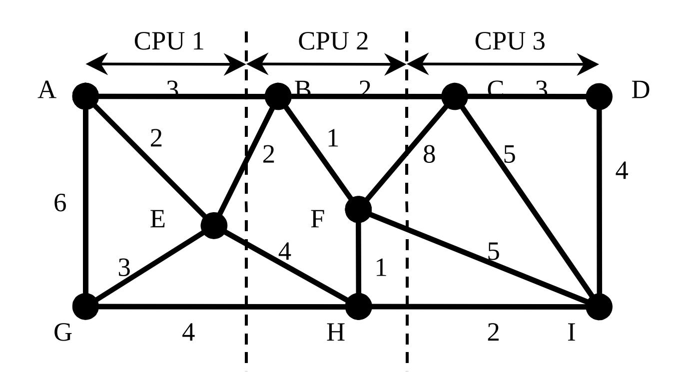
- Los arcos que van de una subgráfica a la otra representan el tráfico en la red.

- Cada subgráfica es una unidad de asignación.

- El algoritmo debe buscar *unidades de asignación fuertemente acopladas:*

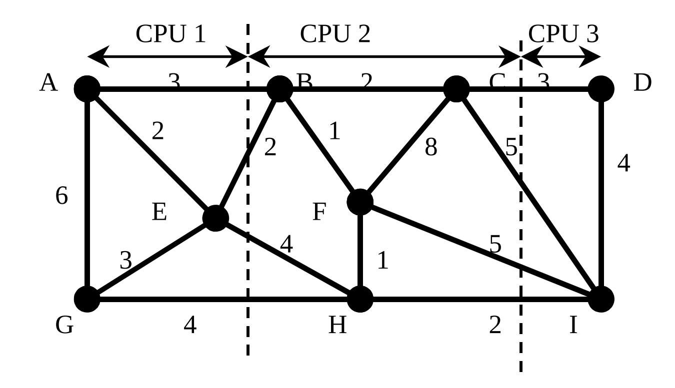
  - Tráfico intenso dentro de la unidad de asignación.
  - Tráfico escaso entre unidades de asignación.

[1] Ver Figura 10.1 de la página 328 y Figura 10.2 de la página 328 [25, Tanenbaum].



EL TRAFICO TOTAL EN LA RED ES LA SUMA DE LAS UNIDADES DE TRAFICO DE LOS ARCOS INTERSECTADOS POR LAS LINEAS PUNTEADAS:
3 + 2 + 4 + 4 + 2 + 8 + 5 + 2 = 30 UNIDADES

Figura 10.1: Una forma de asignar 9 procesos a 3 procesadores.



EL TRAFICO TOTAL EN LA RED ES LA SUMA DE LAS UNIDADES DE TRAFICO DE LOS ARCOS INTERSECTADOS POR LAS LINEAS PUNTEADAS:
3 + 2 + 4 + 4 + 3 + 5 + 5 + 2 = 28 UNIDADES

Figura 10.2: Otra forma de asignar 9 procesos a 3 procesadores.

10.14.2 Un Algoritmo Centralizado

Es un *algoritmo heurístico* que a diferencia del anterior *no precisa información anticipadamente [25, Tanenbaum].*

Es un *algoritmo arriba-abajo (Mutka y Livny) centralizado* porque un *coordinador* mantiene una *tabla de usos:*

- Contiene una entrada por estación de trabajo inicializada en *"0".*
- Cuando ocurren eventos significativos se envían al coordinador mensajes para actualizar la tabla.
- Las *decisiones de asignación* se basan en la tabla:
  - Se toman cuando ocurren eventos de planificación, tales como: se realiza una solicitud, se libera un procesador, el reloj produce una marca de tiempo.
- No se intenta maximizar el uso de la cpu.
- Se procura otorgar a cada usuario una *parte justa* del poder de cómputo.
- Cuando la máquina donde se crea un proceso decide que se debe ejecutar en otra parte:
  - Le pide al coordinador de la tabla de usos que le asigne un procesador:
    - Si existe uno disponible y nadie más lo desea, se otorga el permiso.
    - Si no, la solicitud se niega y se registra.
- Si un usuario ejecuta procesos en máquinas de otros usuarios acumula puntos de penalización por segundo, lo que se registra en la tabla de usos.
- Si un usuario tiene solicitudes pendientes insatisfechas, se restan puntos de penalización.
- Si no existen solicitudes pendientes y ningún procesador está en uso, la entrada de la tabla de usos se desplaza un cierto número de puntos hacia el *"0"*, hasta alcanzarlo.
- El movimiento de puntos hacia arriba y abajo da nombre al algoritmo.

Un *puntaje positivo* en una entrada de la *tabla de usos* indica que la *estación de trabajo* relacionada es un *usuario de los recursos del sistema.*

Un *puntaje negativo* significa que *precisa recursos.*

Una *puntuación "0"* es *neutra.*

La *heurística* utilizada para la *asignación de procesadores* es la siguiente:

- Cuando un procesador se libera gana la solicitud pendiente cuyo poseedor tiene la puntuación menor.
- Un usuario que no ocupe procesadores y que tenga pendiente una solicitud durante mucho tiempo:
  - Siempre vencerá a alguien que utilice muchos procesadores.
  - Se cumple con el principio de asignar la capacidad de manera justa.

10.14.3 Un Algoritmo Jerárquico

El algoritmo anterior no se adapta bien a los *sistemas de gran tamaño [25, Tanenbaum]*, pues el *nodo central* se convierte en un cuello de botella y en un *único punto de fallo.*

Una solución son los *algoritmos jerárquicos* que:

- Mantienen la sencillez de los centralizados.
- Se escalan mejor que los centralizados.

Un método consiste en *organizar a los procesadores en jerarquías lógicas* independientes de la estructura física:

- Se establece un árbol jerárquico con distintos niveles.
- Para cada grupo de máquinas hay una máquina administradora:
  - Mantiene un registro de las máquinas ocupadas y las inactivas.
- Cada procesador se comunica con un superior y un número reducido de subordinados:
  - El flujo de información es controlable.

En caso de falla de un equipo con funciones jerárquicas:

- Lo puede reemplazar un subordinado:
  - La elección la pueden hacer los subordinados, los pares jerárquicos del equipo fallado o el superior jerárquico del mismo.

Para disminuir la vulnerabilidad se puede tener en la cima del árbol jerárquico no uno sino un grupo de equipos; si alguno del grupo falla los restantes eligen a un subordinado para integrar el grupo superior.

Las tareas se pueden crear en cualquier parte de la jerarquía y pueden requerir varios procesos, es decir varios procesadores.

Cada *administrador* debe mantener un *registro de sus equipos dependientes que estén disponibles.*

Si el administrador que recibe una solicitud determina que no tiene suficientes procesadores disponibles, transfiere la solicitud hacia arriba a su superior, quien también podría trasladarla hacia arriba nuevamente.

Si el administrador determina que sí puede satisfacer la solicitud:

- Divide la solicitud en partes y la distribuye a los administradores subordinados a él.
- Los subordinados repiten esta operación hasta llegar al nivel inferior.
- Los procesadores se señalan como *"ocupados"* y el número de procesadores asignados se informa hacia arriba.

Un *importante problema* consiste en que podría haber *varias solicitudes en distintas etapas del algoritmo de asignación:*

- Puede conducir a estimaciones no actualizadas del número de procesadores disponibles (también pudieron salir de servicio algunos de los considerados disponibles).
- Podrían presentarse situaciones de competencia, bloqueo, etc. en el intento de asignación de procesadores.

10.14.4 Un Algoritmo Distribuido Heurístico (Eager)

Al crearse un proceso [25, Tanenbaum]:

- La máquina donde se origina envía mensajes de prueba a una máquina elegida al azar; pregunta si su carga está por debajo de cierto valor de referencia.
- Si la respuesta es positiva el proceso se envía a ese lugar.
- Si no, se elige otra máquina para la prueba.
- Luego de *"n"* pruebas negativas el algoritmo termina y el proceso se ejecuta en la máquina de origen.

10.14.5 Un Algoritmo de Remates

Utiliza un *modelo económico* con [25, Tanenbaum]:

- Compradores y vendedores de servicios.
- Precios establecidos por la oferta y la demanda.

Los *procesos* deben *comprar tiempo de cpu.*

Cada procesador anuncia su precio mediante un archivo que todos pueden leer (es el precio pagado por el último cliente).

Los distintos procesadores pueden tener distintos precios según sus características y servicios.

Cuando un *proceso* desea iniciar un *proceso hijo:*

- Verifica si alguien ofrece el servicio que necesita.
- Determina el conjunto de procesadores que pueden prestar sus servicios.
- Selecciona el mejor candidato según precio, rapidez, relación precio / desempeño, tipo de aplicación, etc.
- Genera una oferta y la envía a su primer opción.

Los *procesadores:*

- Reúnen las ofertas recibidas y eligen una.
- Informan a los ganadores y perdedores.
- Ejecutan los procesos.
- Actualizan los precios.

10.15 Planificación en Sistemas Distribuidos

Generalmente cada procesador hace su planificación local (si tiene varios procesos en ejecución) independientemente de lo que hacen los otros procesadores [25, Tanenbaum].

La *planificación independiente no es eficiente* cuando se ejecutan en distintos procesadores un *grupo de procesos:*

- Relacionados entre sí.
- Con una gran interacción entre los procesos.

Se necesita una forma de *garantizar que los procesos con comunicación frecuente se ejecuten de manera simultánea.*

En muchos casos un grupo de procesos relacionados entre sí iniciarán juntos.

La *comunicación dentro de los grupos* debe *prevalecer* sobre la *comunicación entre los grupos.*

Se debe disponer de un número de procesadores suficiente para soportar al grupo de mayor tamaño.

Cada procesador se multiprograma con *"n"* espacios para los procesos *(multiprogramación de nivel "n").*

El *algoritmo de Ousterhout* utiliza el concepto de *coplanificación:*

- Toma en cuenta los *patrones de comunicación entre los procesos* durante la planificación.
- Debe garantizar que todos los miembros del grupo se ejecuten *al mismo tiempo.*
- Se emplea una matriz conceptual donde:
  - Las filas son espacios de tiempo.
  - Las columnas son las tablas de procesos de los procesadores.
- Cada *procesador* debe utilizar un *algoritmo de planificación Round Robin:*
  - Todos los procesadores ejecutan el proceso en el espacio *"0"* durante un cierto período fijo.
  - Todos los procesadores ejecutan el proceso en el espacio *"1"* durante un cierto período fijo, etc.
- Se deben mantener *sincronizados* los intervalos de tiempo.
- Todos los *miembros de un grupo* se deben colocar en el *mismo número de espacio de tiempo* pero en *procesadores distintos.*

Capítulo 11

Sistemas Distribuidos de Archivos

11.1 Introducción a los Sistemas Distribuidos de Archivos

Muchos aspectos son *similares a los de los sistemas convencionales centralizados* [25, Tanenbaum].

En un sistema distribuido es importante distinguir entre los conceptos de *servicio de archivos* y el *servidor de archivos.*

El servicio de archivos:

- Es la especificación de los servicios que el sistema de archivos ofrece a sus clientes.
- Describe las primitivas disponibles, los parámetros que utilizan y las acciones que llevan a cabo.
- Define precisamente el servicio con que pueden contar los clientes sin decir nada respecto de su implantación.

El despachador (servidor) de archivos:

- Es un proceso que se ejecuta en alguna máquina y ayuda con la implantación del servicio de archivos.
- Puede haber uno o varios en un sistema.
- Los clientes no deben ser conscientes de la forma de implantar el sistema de archivos:
  - No precisan conocer el número de servidores de archivos, su posición o función.
  - Deberían ver al sistema distribuido de archivos como un sistema de archivos normal de uniprocesador.

Generalmente un *servidor de archivos* es un proceso del usuario (a veces del núcleo) que se ejecuta en una máquina:

- Un sistema puede contener varios servidores de archivos, cada uno con un servicio distinto:
  - Ej.: un sistema con un servidor de archivos en "UNIX" y otro en "DOS".
  - Cada proceso usuario utilizaría el servidor apropiado.

11.2 Diseño de los Sistemas Distribuidos de Archivos

Los *componentes* de un sistema distribuido de archivos son [25, Tanenbaum]:

- El verdadero *servicio de archivos:*
  - Realiza operaciones en los archivos individuales: lectura, escritura, adición.
- El *servicio de directorios:*
  - Crea y maneja directorios, añade y elimina archivos de los directorios, etc.

11.3 La Interfaz del Servicio de Archivos

La *protección en los sistemas distribuidos* utiliza *las mismas técnicas* de los sistemas con uniprocesador [25, Tanenbaum]:

- *Posibilidades:*
  - Cada usuario tiene un permiso o posibilidad para cada objeto al que tiene acceso:
    * Determina los *tipos de accesos* permitidos.
- *Listas para control de acceso:*
  - Se asocia a cada archivo una lista implícita o explícita de:
    * Los *usuarios* que pueden tener acceso al archivo.
    * Los *tipos de acceso* permitidos a cada uno de ellos.

Los *servicios de archivos* se pueden clasificar en dos tipos:

- *Modelo carga / descarga:*
  - Las principales operaciones son la lectura de un archivo y la escritura en un archivo.
  - La lectura transfiere todo un archivo de uno de los servidores de archivos al cliente solicitante.
  - La escritura transfiere en sentido contrario.
  - Los archivos se pueden almacenar en memoria o en un disco local.
- *Modelo de acceso remoto:*
  - El sistema de archivos se ejecuta con todas las funciones en los servidores y no en los clientes.

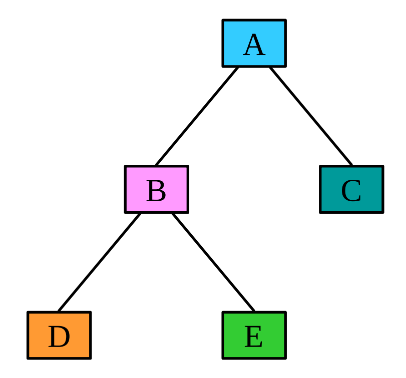


Figura 11.1: Arbol de directorios contenido en una máquina.

11.4 La Interfaz del Servidor de Directorios

Proporciona operaciones para crear y eliminar directorios, nombrar y cambiar el nombre de archivos y mover archivos de un directorio a otro [25, Tanenbaum].

Se utiliza un *sistema jerárquico de archivos*, representado por un *árbol de directorios.*[1]

En ciertos sistemas es posible crear *enlaces o apuntadores a un directorio arbitrario:*

- Se pueden colocar en cualquier directorio.
- Se pueden construir gráficas de directorios.

En una *jerarquía con estructura de árbol* solo se puede eliminar un enlace con un directorio si el directorio al cual se apunta está vacío.

En una *gráfica* se permite la eliminación de un enlace mientras exista al menos otro:

- Se utiliza un contador de referencias para determinar si el enlace por eliminar es el último.
- Se puede armar una gráfica de directorios comprendiendo a directorios de dos o más máquinas.
- La eliminación de enlaces puede llevar a directorios y archivos a la condición de huérfanos, es decir que no pueden ser alcanzados desde el directorio raíz.[2]

Un *aspecto fundamental de diseño* en sistemas distribuidos es *si todas las máquinas y procesos tendrán exactamente la misma visión de la jerarquía de los directorios.*

En los sistemas que utilizan *varios servidores de archivos* mediante el montaje remoto generalmente los diversos clientes tienen una *visión diferente del sistema de archivos*, pero la *desventaja* es que el sistema *no se comporta como un único sistema de tiempo compartido.*

Una cuestión relacionada es si existe un *directorio raíz global* al que todas las máquinas reconozcan como la raíz; una posibilidad es que la raíz solo contenga una entrada por cada servidor.

[1]Ver Figura 11.1 de la página 335 [25, Tanenbaum].
[2]Ver Figura 11.2 de la página 336 [25, Tanenbaum].

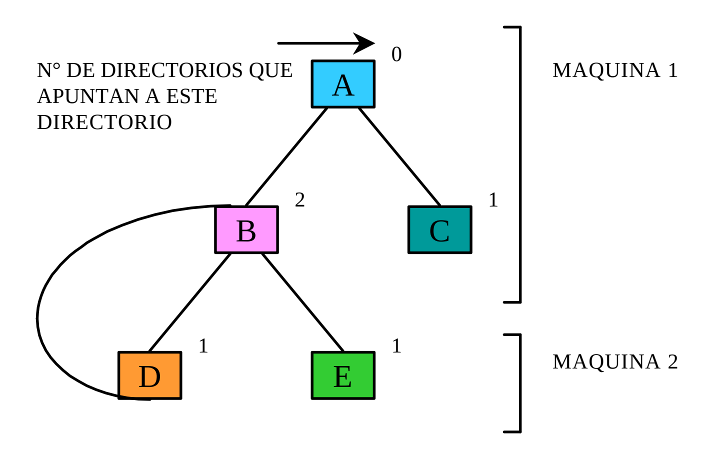


Figura 11.2: Gráfica de directorios de dos máquinas.

11.4.1 Transparencia de los Nombres

La *transparencia con respecto a la posición* significa que el nombre de la ruta de acceso no sugiere la posición del archivo:

- Se individualiza al servidor pero no se indica dónde está, por ello puede moverse dentro de la red sin necesidad de cambiar la ruta.

- Ej.: */servidor1/dir1/dir2/x.*

Si el primer componente de todas las rutas de acceso es el servidor, el sistema no puede desplazar el archivo a otro servidor en forma automática porque cambiaría el nombre de la ruta de acceso.

Un sistema donde los archivos se pueden desplazar sin que cambien sus nombres tiene ***independencia con respecto a la posición****.*

Resumiendo, los *métodos usuales para nombrar los archivos y directorios* en un sistema distribuido son:

- Nombre máquina + ruta de acceso.

- Montaje de sistemas de archivos remotos en la jerarquía local de archivos.

- Un único espacio de nombres que tenga la misma apariencia en todas las máquinas.

11.5 Semántica de los Archivos Compartidos

Cuando se *comparten archivos* es necesario *definir con precisión la semántica* de la lectura y escritura.

En sistemas *monoprocesador* que permiten a los procesos *compartir archivos* (ej.: UNIX) *la semántica generalmente establece:*

- Si un *read* sigue a un *write*, *read* debe regresar el valor recién escrito.

- Si dos *write* se realizan en serie y luego se ejecuta un *read*, el valor que se debe regresar es el almacenado en la última escritura.
- Este modelo se denomina *semántica de UNIX.*

En un *sistema distribuido* la *semántica de UNIX* se puede lograr fácilmente si:

- Solo existe un servidor de archivos.
- Los clientes no ocultan los archivos.

Un problema que se puede presentar se debe a los *retrasos en la red:*

- Si un *read* ocurrido después de un *write* llega primero al servidor obtendrá el valor previo al *write.*

Otro problema es el *desempeño* pobre de un sistema distribuido en donde todas las solicitudes de archivos deben pasar a un *único servidor:*

- Una solución es permitir a los cliente mantener copias locales de los archivos de uso frecuente en sus cachés particulares, lo que ocasiona el siguiente problema:
  - Un cliente modifica localmente un archivo en su caché.
  - Luego otro cliente lee el archivo del servidor.
  - El segundo cliente obtendrá un archivo obsoleto.
- Una solución sería propagar inmediatamente todas las modificaciones de los archivos en caché de regreso al despachador:
  - Resulta prácticamente ineficiente.

Otra *solución* es *relajar la semántica de los archivos compartidos:*

- Los cambios a un archivo abierto solo pueden ser vistos en un principio por el proceso (o tal máquina) que modifico el archivo.
- Los cambios serán visibles a los demás procesos (o máquinas) solo cuando se cierre el archivo y sea actualizado en el servidor.
- Esta regla se conoce como la *semántica de sesión.*

Un *problema* se presenta cuando *dos o más clientes ocultan y modifican el mismo archivo en forma simultánea:*

- Una solución es que al cerrar cada archivo su valor se envía de regreso al servidor:
  - El resultado final depende de quién lo cierre más rápido.

Otro *problema* consiste en que *no se pueden compartir los apuntadores* que para cada archivo indican en la semántica UNIX la posición actual en el archivo.

Un método distinto es que *todos los archivos sean inmutables:*

- No se puede abrir un archivo para escribir en él.
- Solo se permiten las operaciones *create* y *read.*
- Los directorios sí se pueden actualizar.
- Se puede crear un archivo nuevo e introducirlo en el directorio con el nombre de un archivo ya existente:
  - Este se vuelve inaccesible con el mismo nombre.
  - Persiste el problema de cómo tratar la situación presentada cuando dos procesos intentan reemplazar el mismo archivo a la vez.

Otra *vía de solución* para el uso de archivos compartidos en un sistema distribuido es usar las *transacciones atómicas:*

- Se garantiza que todas las llamadas contenidas en la transacción se llevarán a cabo en orden.
- No habrá interferencias de otras transacciones concurrentes.

11.6 Implantación de un Sistema Distribuido de Archivos

La implantación de un sistema distribuido de archivos *incluye* aspectos tales como [25, Tanenbaum]:

- *El uso de los archivos.*
- *La estructura del sistema.*
- *El ocultamiento.*
- *La duplicación o réplica.*
- *El control de la concurrencia.*

11.7 Uso de Archivos

Antes de implantar un sistema de archivos resulta de interés analizar los *"patrones de uso"* de dichos archivos [25, Tanenbaum].

Para determinar los *patrones de uso* es necesario tomar *mediciones* que pueden ser:

- *Estáticas:*
  - Representan una toma instantánea del sistema en un momento dado.

  - Comprenden la distribución de tamaño de los archivos, la distribución de tipo de archivos, la cantidad de espacio que ocupan los archivos de varios tamaños y tipos, etc.

- *Dinámicas:*
  - Registran en una bitácora todas las operaciones que modifican el sistema de archivos.
  - Comprenden información sobre la frecuencia relativa de varias operaciones, el número de archivos abiertos en un momento dado, la cantidad de archivos compartidos, etc.

Las *principales propiedades* observadas son:

- La mayoría de los archivos son pequeños.
- La lectura es más común que la escritura.
- La mayoría de los accesos es secuencial.
- La mayoría de los archivos son de corta vida.
- Es poco usual compartir archivos.
- Los procesos promedio utilizan pocos archivos.
- Distintas clases de archivos poseen propiedades distintas.

11.8 Estructura del Sistema

En ciertos sistemas no existe *distinción entre un cliente y un servidor [25, Tanenbaum]:*

- Todas las máquinas ejecutan el mismo software básico.
- Una máquina que desee dar servicio de archivos lo puede hacer:
  - Debe *exportar* los nombres de los directorios seleccionados, para que otras máquinas los puedan acceder.

En otros sistemas el *servidor de archivos* y el de *directorios* son solo *programas del usuario*, y se puede configurar un sistema para que ejecute o no el *software de cliente o servidor* en la misma máquina.

Los *clientes* y *servidores* también podrían ser *máquinas totalmente distintas* en términos de hardware o de software.

Un aspecto de *implantación* en donde difieren los sistemas es la *forma de estructurar el servicio a directorios y archivos*; las principales *opciones* son las siguientes:

- *Combinar el servicio a directorios y archivos* en un único servidor que administre todas las llamadas a directorios y archivos.

- *Separar el servicio a directorios y archivos* utilizando un servidor de directorios y un servidor de archivos.

Si se considera el caso de *servidores de archivos y directorios independientes:*

- El cliente envía un nombre simbólico al servidor de directorios.
- El servidor de directorios regresa el nombre en binario (ej.: *máquina + nodo_ i*) que comprende el servidor de archivos.
- Es posible que una jerarquía de directorios se reparta entre varios servidores.
- El servidor que recibe un nombre binario que se refiere a otro servidor puede:
  - Indicar al cliente el servidor que tiene el archivo buscado, para que el cliente lo busque.
  - Enviar la solicitud al siguiente servidor y no contestar.

Un *aspecto estructural* a considerar es *si los servidores de archivos, directorios o de otro tipo deben contener la información de estado de los clientes.*

Una posibilidad es que *los servidores no deben contener los estados*, deben ser *sin estado:*

- Cuando un cliente envía una solicitud a un servidor:
  - El servidor la lleva a cabo, envía la respuesta y elimina de sus tablas internas toda la información relativa a esa solicitud.
  - El servidor no guarda información relativa a los clientes entre las solicitudes.

Otra posibilidad es que *los servidores conserven información de estado de los clientes* entre las solicitudes.

Aclaración:

- Luego de abrir un archivo el servidor debe mantener la información que relacione los clientes con los archivos abiertos por éstos.
- Al abrir un archivo el cliente recibe un descriptor de archivo que se utiliza en las llamadas posteriores para identificación del archivo.
- Al recibir una solicitud el servidor utiliza el descriptor de archivo para determinar el archivo necesario.
- *La tabla que asocia los* ***descriptores de archivo*** *con los* ***archivos*** *propiamente dichos es* ***información de estado****.*

En un *servidor sin estado* cada solicitud debe ser *autocontenida:*

- Debe incluir el nombre del archivo y toda la información para que el servidor realice el trabajo.

- La longitud del mensaje es mayor.

Si un *servidor con estado falla* y sus tablas se pierden:

- Al volver a arrancar no tiene información sobre la relación entre los clientes y los archivos abiertos por éstos.
- La recuperación queda a cargo de los clientes.

Los servidores sin estado tienden a ser más tolerantes de los fallos que los servidores con estados.

11.9 Ocultamiento

En un *sistema cliente - servidor*, cada uno con su memoria principal y un disco, existen cuatro *lugares donde se pueden almacenar los archivos o partes de ellos [25, Tanenbaum]:*

- El disco del servidor.
- La memoria principal del servidor.
- El disco del cliente (si existe).
- La memoria principal del cliente.

Si los archivos se almacenan en el *disco del servidor:*

- Disponen de abundante espacio.
- Serían accesibles a todos los clientes.
- No habrá problemas de consistencia al existir solo una copia de cada archivo.
- Puede haber problemas de desempeño:
  - Antes de que un cliente pueda leer un archivo se lo debe transferir:
    * Del disco del servidor a la memoria principal del servidor.
    * De la memoria principal del servidor a la memoria principal del cliente, a través de la red.
- Se puede mejorar el desempeño ocultando (conservando) los archivos de más reciente uso en la memoria principal del servidor:
  - Un cliente que lea un archivo ya presente en el caché del servidor elimina la transferencia del disco.
  - Se necesita un algoritmo para determinar los archivos o partes de archivos que deben permanecer en el caché.

El algoritmo debe *resolver los siguientes problemas:*

- La unidad que maneja el caché.

- Qué hacer si se utiliza toda la capacidad del caché y hay que eliminar a alguien.

Respecto de la *unidad que maneja el caché:*

- Puede manejar *archivos completos* o *bloques del disco.*

- El ocultamiento de archivos completos que se pueden almacenar en forma adyacente en el disco permite un buen desempeño en general.

- El ocultamiento de bloques de disco utiliza el caché y el espacio en disco más eficientemente.

Respecto de *qué hacer cuando se utiliza toda la capacidad del caché* y hay que eliminar a alguien:

- Se puede utilizar cualquier *algoritmo de ocultamiento*, por ej.: *LRU* mediante listas ligadas.

- Cuando hay que eliminar a alguien de la memoria:

  - Se elige al más antiguo.
  - Si existe una copia actualizada en el disco se descarta la copia del caché.
  - De lo contrario primero se actualiza el disco.

El mantenimiento de un *caché en la memoria principal del servidor* es fácil de lograr y es totalmente *transparente a los clientes.*
Si se utiliza *ocultamiento en el lado del cliente:*

- Se elimina el acceso a la red para transferir del servidor al cliente.

- El disco del cliente generalmente es más lento y de menor capacidad.

- Generalmente es más rápido y más sencillo tener un caché en la memoria principal del servidor que en el disco del cliente.

Si el *caché se coloca en la memoria principal del cliente* las principales opciones son:

- Ocultar los archivos dentro del propio espacio de direcciones de un proceso de usuario.

- Colocar el caché en el núcleo.

- Ocultar el caché en un proceso manejador del caché, independiente y a nivel usuario.

11.9.1 Consistencia del Caché

El ocultamiento por parte del cliente introduce ***inconsistencia*** *en el sistema.*

Si dos clientes leen un mismo archivo en forma simultánea y después lo modifican, aparecen algunos problemas:

- Cuando un tercer proceso lee el archivo del servidor obtendrá la versión original y no alguna de las nuevas:
  - Se puede evitar mediante la *"semántica de sesión":*
  - No es aplicable cuando se requiere la *"semántica de UNIX".*
- Cuando dos archivos se escriben de nuevo al servidor, el último de ellos se escribirá sobre el otro.

Una *solución a la inconsistencia del caché* es el *algoritmo de escritura a través del caché:*

- Cuando se modifica una entrada del caché (archivo o bloque), el nuevo valor:
  - Se mantiene dentro de él.
  - Se envía de inmediato al servidor.

Los principales problemas de la *escritura a través del caché* son los siguientes:

- Posible suministro de valores obsoletos:
  - Un proceso cliente en la máquina *"A"* lee un archivo *"f"* y mantiene a *"f"* en su caché.
  - Un cliente en la máquina *"B"* lee el mismo archivo, lo modifica y lo escribe en el servidor.
  - Otro proceso cliente inicia en la máquina *"A"* abriendo y leyendo *"f"*, que se toma del caché.
  - El valor de *"f"* es obsoleto.
  - Una solución consiste en exigir al manejador del caché que verifique el servidor antes de proporcionar al cliente un archivo del caché:
    - Generalmente utilizará una *"RPC"* y poca información de control.
- El tráfico en la red en el caso de las escrituras es igual que en el caso de no ocultamiento:
  - Para mejorar se puede aplicar el siguiente procedimiento de *retraso en la escritura:*
    - En vez de ir hacia el servidor en el instante en que se realiza la escritura, el cliente:
      - Hace una notificación de que ha actualizado un archivo.

      · Cada cierto intervalo (ej.: 30") todas las actualizaciones se agrupan y envían la servidor al mismo tiempo (un bloque).

  - El retraso en la escritura obscurece la semántica:

    * Si otro proceso lee el archivo, el resultado dependerá de la sincronización de los eventos.

Otro algoritmo para manejar el *caché de archivos del cliente* es el de *escritura al cierre:*

- Se adopta la semántica de sesión.

- Solo se escribe un archivo nuevamente en el servidor cuando el archivo se cierra:

  - Se podría esperar (ej.: 30") para ver si el archivo es eliminado en ese lapso.

Un método distinto a la consistencia es utilizar un *algoritmo de control centralizado:*

- Al abrir un archivo la máquina envía un mensaje al servidor para anunciar este hecho.

- El servidor de archivos tiene un registro de los archivos abiertos, sus poseedores y si están abiertos para lectura, escritura o ambos procesos.

- Si se abre un archivo para lectura otros procesos lo pueden abrir para lectura pero no para escritura.

- Si se abre un archivo para escritura se debe evitar abrirlo para lectura desde otro proceso.

- Al cerrar un archivo:

  - Se debe informar al servidor para que actualice sus tablas.
  - Se puede enviar el archivo modificado al servidor.

11.10 Réplica

Frecuentemente los sistemas distribuidos de archivos proporcionan la *réplica de archivos como un servicio [25, Tanenbaum]:*

- Existen varias copias de algunos archivos.

- Cada copia está en un servidor de archivos independiente.

Las *principales razones* para la réplica son:

- Aumentar la confiabilidad al disponer de respaldos independientes de cada archivo.

- Permitir el acceso a archivos aún cuando falle un servidor de archivos.

- Repartir la carga de trabajo entre varios servidores.

Un sistema *es **transparente** con respecto a la réplica* si la misma se administra sin intervención del usuario.

Una forma de llevar a cabo la réplica consiste en que el *programador* controle todo el proceso *(réplica explícita):*

- Los archivos y las copias adicionales se crean en servidores específicos.
- Las direcciones en la red de todas las copias se asocian con el nombre del archivo.

Un *método alternativo* es la *réplica retrasada:*

- Solo se crea una copia de cada archivo en un servidor.
- El servidor crea réplicas en otros servidores, a posteriori, automáticamente y sin intervención del proceso de usuario.

Otro método consiste en el uso de la *comunicación en grupo:*

- Todas las operaciones de escritura se transmiten simultáneamente a todos los servidores.
- Las copias adicionales se hacen al mismo tiempo que el original.

11.10.1 Protocolos de Actualización

*El principal problema es asegurar la **sincronización** de las distintas copias.*

Un *algoritmo* posible es el de *réplica de la copia primaria:*

- Uno de los servidores se denomina como primario.
- Los demás servidores son secundarios.
- La actualización se envía al servidor primario:
  - Realiza los cambios localmente.
  - Envía comandos a los servidores secundarios para ordenarles la misma modificación.
- Las lecturas se pueden hacer de cualquier copia.
- La *desventaja* es que si falla el primario no se pueden llevar a cabo las actualizaciones.

Otro posible *algoritmo* es el del *voto o de Gifford:*

- La idea fundamental es exigir a los clientes que soliciten y adquieran el permiso de varios servidores antes de leer o escribir un archivo replicado.
- Se utiliza el número de versión, que identifica la versión del archivo y es la misma para todos los archivos recién actualizados.

- Para leer un archivo con *"N"* réplicas un cliente debe conformar un *quórum de lectura*, es decir una colección arbitraria de *"N_r"* servidores o más.
- Para modificar un archivo se necesita un *quórum de escritura* de al menos *"N_w"* servidores.
- Se debe cumplir que *"N_r" + "N_w" > "N"*, por lo cual nunca se podrá obtener un quórum de lectura y otro de escritura al mismo tiempo.
- Generalmente *"N_r"* es muy pequeño y *"N_w"* muy cercano a *"N"* ya que generalmente las lecturas son más frecuentes que las escrituras.
- Una *variante* es el *algoritmo del voto con fantasma:*
  - Crea un *servidor fantasma* para cada servidor real fallido:
    - Interviene solo en el quórum de escritura.
    - La escritura solo tiene éxito si al menos uno de los servidores es real.

11.11 Conclusiones Importantes Respecto de la Implantación de un Sistema Distribuido de Archivos

Los principios generalmente considerados fundamentales del diseño de un sistema distribuido de archivos son [25, Tanenbaum]:

- Las *estaciones de trabajo* tienen ciclos que hay que utilizar:
  - Si se tiene la opción de hacer algo en una estación de trabajo o en un servidor:
    - Elegir la estación de trabajo.
    - Los ciclos de cpu de la estación de trabajo son menos costosos que los ciclos de un servidor.
- Utilizar el *caché* el máximo posible:
  - Frecuentemente ahorran considerable:
    - Tiempo de cómputo.
    - Ancho de banda de la red.
- Explotar las *propiedades de uso:*
  - Considerar la posibilidad de implantar tratamientos diferenciales para los archivos transitorios de corta vida y no compartidos.
  - Tener presente la dificultad de habilitar diferentes vías para hacer lo mismo.
  - Considerar aspectos tales como eficiencia y sencillez.
- Minimizar el *conocimiento* y *modificación* a lo largo del sistema:

- Es importante para lograr escalabilidad.
- Generalmente son útiles en este sentido los diseños jerárquicos.

- Confiar en el *menor número* posible de entidades:
  - Se trata de un principio ya establecido en el mundo de la seguridad.

- Crear *lotes de trabajo* mientras sea posible:
  - El uso del procesamiento por lotes puede contribuir a un mejor desempeño.

11.12 Tendencias en los Sistemas Distribuidos de Archivos

Es probable que los *cambios en el hardware* tengan un efecto muy importante en los futuros sistemas distribuidos de archivos [25, Tanenbaum].

También es probable el impacto del *cambio en las expectativas del usuario.*

11.13 Consideraciones Respecto del Hardware

El abaratamiento de la memoria principal permitirá disponer de *servidores con memorias cada vez mayores [25, Tanenbaum]:*

- Se podría alojar directamente en memoria el sistema de archivos logrando mayor sencillez y desempeño.

- Se debería prever la obtención de respaldos continuos o por incrementos ante la posibilidad del corte en el suministro eléctrico.

- El respaldo podría hacerse en discos ópticos regrabables que tengan una asociación uno a uno con la memoria:
  - El byte *"k"* de la memoria correspondería al byte *"k"* del disco.

La disponibilidad de *redes de fibra óptica de alta velocidad* permitiría esquemas tales como:

- Un servidor de archivos en la memoria principal del servidor con respaldo en el disco óptico.

- Eliminación del disco del servidor y del caché del cliente.

- Se simplificaría significativamente el software.

La posible construcción de *interfaces de red especializadas* que permitan resolver por hardware problemas difíciles de soportar por software:

- Cada interfaz de red tendría un mapa de bits con un bit por cada archivo en el caché.

- Se podrían habilitar cerraduras por archivo.

- Para modificar un archivo un procesador activaría el bit correspondiente en la interfaz.

11.14 Escalabilidad

Una tendencia definida en los sistemas distribuidos es *hacia los sistemas cada vez mas grandes [25, Tanenbaum].*

Los sistemas distribuidos de archivos que operan bien para cientos de máquinas podrían fallar en algún aspecto trabajando con *miles o decenas de miles de máquinas.*

Generalmente los algoritmos centralizados no se escalan bien ya que el servidor centralizado podría convertirse en un cuello de botella; por ello se podría separar el sistema en unidades más pequeñas relativamente independientes entre sí.

Las *transmisiones* también son un área problemática:

- Si cada máquina transmite una vez por segundo:
  - Con *"n"* máquinas habría *"n"* transmisiones y *"n^2"* interrupciones por segundo.
  - Si *"n"* crece esto se puede convertir en un problema.

En general los recursos y algoritmos no deben ser lineales con respecto al número de usuarios.

11.15 Redes en un Area Amplia

Generalmente los sistemas distribuidos se asocian con *redes de área local (LAN),* pero cada vez será mayor la necesidad de conectarlos entre sí cubriendo *grandes áreas (nacionales, regionales, continentales, etc.) [25, Tanenbaum].*

Los sistemas de archivos deberán soportar estas necesidades teniendo presente la *heterogeneidad* de los equipos, códigos de representación (ASCII, EBCDIC, etc.), formatos, etc.

Deberá atenderse a los cambios de *tendencia en los requerimientos de las aplicaciones.*

Un problema adicional e inherente en los sistemas distribuidos masivos es el *ancho de banda de la red,* que puede resultar insuficiente para el desempeño esperado.

11.16 Usuarios Móviles

Los usuarios de *equipos móviles (laptop, notebook, etc.)* están gran parte del tiempo *desconectados* del sistema de archivos de su organización [25, Tanenbaum]:

- Requieren una solución, que podría usar ocultamiento:
  - Cuando está conectado el usuario carga al equipo móvil los archivos que cree necesitará después.
  - Los utiliza mientras está desconectado.
  - Al reconectarse, los archivos en el caché deben fusionarse con los existentes en el árbol de directorios, logrando la sincronización.
  - La conexión para la sincronización puede ser problemática si se utiliza un enlace de ancho de banda reducido.

Lo deseable sería un sistema distribuido totalmente transparente para su uso simultáneo por parte de millones de usuarios móviles que frecuentemente se desconecten.

11.17 Tolerancia de Fallos

La difusión de los sistemas distribuidos incrementa la *demanda de sistemas que esencialmente nunca fallen [25, Tanenbaum].*

Los sistemas tolerantes a fallos requerirán cada vez más una considerable *redundancia* en hardware, comunicaciones, software, datos, etc.

La *réplica* de archivos sería un requisito esencial.

También debería contemplarse la posibilidad de que los sistemas funcionen aún con la *carencia de parte de los datos.*

Los tiempos de fallo aceptables por los usuarios serán cada vez menores.

Capítulo 12

Rendimiento

12.1 Introducción a la Medición, Control y Evaluación del Rendimiento

Un sistema operativo es en primer lugar un administrador de recursos, por ello es importante poder *determinar con qué efectividad* administra sus recursos un sistema determinado [7, Deitel].

Generalmente hay un gran *potencial de mejora* en el uso de los recursos existentes, pero:

- Muchas instalaciones realizan muy poco o ningún control y evaluación.
- Cuando se hacen controles específicos se generan grandes cantidades de datos que muchas veces no se sabe cómo interpretar.

Las instalaciones rara vez cuentan con personal versado en las técnicas de *análisis de rendimiento*.

Durante los primeros años del desarrollo de las computadoras el hardware representaba el costo dominante de los sistemas y debido a ello los estudios de rendimiento *se concentraban en el hardware.*

Actualmente y según la tendencia apreciable:

- El software representa una porción cada vez mayor de los presupuestos informáticos.
- El software incluye el S. O. de multiprogramación / multiproceso, sistemas de comunicaciones de datos, sistemas de administración de bases de datos, sistemas de apoyo a varias aplicaciones, etc.
- El software frecuentemente oculta el hardware al usuario creando una máquina virtual, que está definida por las características operativas del software.

Un software deficiente y / o mal utilizado puede ser causa de un rendimiento pobre del hardware, por lo tanto es importante controlar y evaluar el *rendimiento del hardware y del software.*

12.2 Tendencias Importantes que Afectan a los Aspectos del Rendimiento

Con los avances en la tecnología de *hardware* los costos del mismo han *decrecido* drásticamente y todo hace suponer que esta tendencia continuará [7, Deitel].

Los costos de *trabajo (personal)* han ido *aumentando:*

- Significan un porcentaje importante del costo de los sistemas informáticos.
- Se debe reformular el aspecto del rendimiento del hardware base y medirlo de manera más adaptada a la productividad humana.

El advenimiento del *microprocesador* en la década de 1.970:

- Ha permitido bajar considerablemente el costo de los ciclos de cpu.
- Ha desplazado el foco de atención de la evaluación del rendimiento a otras áreas donde los costos no disminuyeron proporcionalmente; ej.: utilización de dispositivos de entrada / salida.

También *influyen* en los puntos de vista sobre la *evaluación del rendimiento* aspectos tales como:

- Construcción de redes.
- Procesamiento distribuido.

Las conexiones se hacen con *redes* y no solo con computadoras específicas:

- Se puede disponer de cientos o miles de *sistemas de computación.*
- Se puede acceder a complejos *sistemas de comunicaciones de datos [21, Stallings].*

12.3 Necesidad del Control y de la Evaluación del Rendimiento

Los *objetivos corrientes en la evaluación del rendimiento* generalmente son [7, Deitel]:

- *Evaluación de selección:*
  - El evaluador debe decidir si la adquisición de un sistema de computación es apropiada.
- *Proyección del rendimiento:*
  - El evaluador debe estimar el rendimiento de un:
    - Sistema inexistente.
    - Nuevo sistema.

        * Nuevo componente de hardware o de software.

- *Control del rendimiento:*
  - El evaluador acumula datos del rendimiento de un sistema o componente existente para:
    * Asegurar que el sistema cumple con sus metas de rendimiento.
    * Ayudar a estimar el impacto de los cambios planeados.
    * Proporcionar los datos necesarios para tomar decisiones estratégicas.

En las *primeras fases del desarrollo* de un nuevo sistema se intenta predecir:

- La naturaleza de las aplicaciones que correrán en el sistema.
- Las cargas de trabajo que las aplicaciones deberán manejar.

Durante el *desarrollo e implementación* de un nuevo sistema se intenta determinar:

- La mejor organización del hardware.
- Las estrategias de administración de recursos que deberán implantarse en el S. O.
- Si el sistema cumple o no con sus objetivos de rendimiento.

Frecuentemente son necesarios *procesos de configuración* de los sistemas para que puedan servir a las necesidades.

Los *procesos de sintonización* del sistema tienden a mejorar el rendimiento en base a ajustar el sistema a las características de la instalación del usuario.

12.4 Mediciones del Rendimiento

El rendimiento expresa la manera o la eficiencia con que un sistema de computación cumple sus metas [7, Deitel].

El rendimiento es una cantidad *relativa más que absoluta* pero suele hablarse de *medidas absolutas de rendimiento,* ej.: número de trabajos atendidos por unidad de tiempo.

Algunas mediciones son *difíciles de cuantificar*, ej.: facilidad de uso.

Otras mediciones son *fáciles de cuantificar*, ej.: accesos a un disco en la unidad de tiempo.

Las *mediciones de rendimiento* pueden estar:

- Orientadas hacia el usuario, ej.: tiempos de respuesta.
- Orientadas hacia el sistema, ej.: utilización de la cpu.

Algunas *mediciones del rendimiento comunes* son:

- *Tiempo de regreso:*

- Tiempo desde la entrega del trabajo hasta su regreso al usuario (para procesamiento por lotes).

- *Tiempo de respuesta:*

  - Tiempo de regreso de un sistema interactivo.

- *Tiempo de reacción del sistema:*

  - Tiempo desde que el usuario presiona *"enter"* hasta que se da la primera sección de tiempo de servicio.

Las anteriores son *cantidades probabilísticas* y se consideran como *variables aleatorias* en los estudios de:

- Simulación.
- Modelado de sistemas.

Otras medidas del rendimiento utilizadas son:

- *Varianza de los tiempos de respuesta* (o de otra de las variables aleatorias consideradas):

  - Es una medida de dispersión.
  - Si es pequeña indica tiempos próximos a la media.
  - Si es grande indica tiempos alejados de la media.
  - Es una medida de la *predecibilidad.*

- *Capacidad de ejecución:*

  - Es la medida de la ejecución de trabajo por unidad de tiempo.

- *Carga de trabajo:*

  - Es la medida de la cantidad de trabajo que:
    * Ha sido introducida en el sistema.
    * El sistema debe procesar normalmente para funcionar de manera aceptable.

- *Capacidad:*

  - Es la medida de la capacidad de rendimiento máxima que un sistema puede tener siempre que:
    * El sistema esté listo para aceptar más trabajos.
    * Haya alguno inmediatamente disponible.

- *Utilización:*

- Es la fracción de tiempo que un recurso está en uso.
- Es deseable un gran porcentaje de utilización pero éste puede ser el resultado de un uso ineficiente.
- Cuando se aplica a la cpu se debe distinguir entre:
  * Uso en trabajos productivos de aplicación.
  * Uso en sobrecarga del sistema.

12.5 Técnicas de Evaluación del Rendimiento

Tiempos

Los tiempos proporcionan los medios para realizar comparaciones rápidas del hardware [7, Deitel].

Una posible *unidad de medida* es el *"mips":* millón de instrucciones por segundo.

Los tiempos se usan para comparaciones rápidas; se utilizan operaciones básicas de hardware.

Mezclas de instrucciones

Se usa un *promedio ponderado* de varios tiempos de las instrucciones más apropiadas para una aplicación determinada; los equipos pueden ser comparados con mayor certeza de la que proporcionan los tiempos por sí solos.

Son útiles para comparaciones rápidas del hardware.

Programas del núcleo

Un programa núcleo es un *programa típico* que puede ser ejecutado en una instalación.

Se utilizan los tiempos estimados que suministran los fabricantes para cada máquina para calcular su tiempo de ejecución.

Se corre el programa típico en las distintas máquinas para obtener su tiempo de ejecución.

Pueden ser útiles para la evaluación de ciertos componentes del software, por ej. compiladores; pueden ayudar a determinar qué compilador genera el código más eficiente.

Modelos analíticos

Son representaciones matemáticas de sistemas de computación o de componentes de sistemas de computación.

Generalmente se utilizan los *modelos de:*

- *Teoría de colas.*
- *Procesos de Markov.*

Requieren un *gran nivel matemático del evaluador* y son *confiables solo en sistemas sencillos*, ya que en sistemas complejos los supuestos simplificadores pueden invalidar su utilidad y aplicabilidad.

Puntos de referencia (o programas de comparación del rendimiento)

Son *programas reales* que el evaluador ejecuta en la máquina que se está evaluando.

Generalmente es un *programa de producción:*

- Típico de muchos trabajos de la instalación.

- Que se ejecuta con regularidad.

El programa completo se ejecuta *en la máquina real con datos reales.*

Se deben seleccionar cuidadosamente los *puntos de referencia* para que sean representativos de los trabajos de la instalación.

Los efectos del software pueden experimentarse directamente en vez de estimarse.

Programas sintéticos

Combinan las técnicas de los núcleos y los puntos de referencia.

Son programas reales diseñados para ejercitar características específicas de una máquina.

Simulación

Es una técnica con la cual el evaluador desarrolla un *modelo computarizado del sistema* que se está evaluando.

Es posible preparar un modelo de un sistema inexistente y ejecutarlo para ver cómo se comportaría en ciertas circunstancias; se puede evitar la construcción de sistemas mal diseñados.

Los *simuladores* son muy aplicados en las industrias espacial y de transportes.

Los *simuladores* pueden ser:

- *Manejados por eventos:*
  - Son controlados por los eventos producidos en el simulador según distribuciones probabilísticas.
- *Manejados por libreto:*
  - Son controlados por datos obtenidos de forma empírica y manipulados cuidadosamente para reflejar el comportamiento anticipado del sistema simulado.

Control del rendimiento

Es la *recolección y análisis de información* relativa al rendimiento del sistema existente.

Permite localizar embotellamientos con rapidez.

Puede ayudar a decidir la forma de mejorar el rendimiento.

Puede ser útil para determinar la distribución de trabajos de varios tipos; permitiría aconsejar el uso de *compiladores optimizadores* o *compiladores rápidos y sucios.*

El control del rendimiento puede hacerse por medio de técnicas de hardware o de software.

Los monitores de software:

- Generalmente son económicos.
- Pueden distorsionar las lecturas del rendimiento debido a que consumen recursos del sistema.

Los monitores de hardware:

- Generalmente son más costosos.

- Su influencia sobre la operación del sistema es mínima.

Los monitores:

- Producen grandes cantidades de datos que deben ser analizados manualmente o por sistema.
- Indican con precisión cómo está funcionando un sistema.
- Son de mucha ayuda para evaluar sistemas en desarrollo y tomar las decisiones de diseño adecuadas.

Los *rastreos de ejecución de instrucciones (trace)* o *rastreos de ejecución de módulos* pueden revelar embotellamientos.

Un *rastreo de ejecución de módulos* puede mostrar que se está ejecutando un pequeño subconjunto de módulos durante gran parte del tiempo:

- Los diseñadores deberán optimizarlos para mejorar en gran medida el rendimiento del sistema.
- Se podría eliminar el costo de optimización de los módulos poco usados.

12.6 Embotellamientos y Saturación

Los *recursos* administrados por los S. O. *se acoplan e interactúan* de maneras complejas para afectar al total de la operación del sistema [7, Deitel].

Ciertos *recursos* pueden sufrir *embotellamientos* que limitan el rendimiento del sistema:

- No pueden realizar su parte del trabajo.
- Otros recursos pueden estar con exceso de capacidad.

Un *embotellamiento tiende a producirse en un recurso* cuando el tráfico de trabajos o procesos de ese recurso comienza a alcanzar su *capacidad límite:*

- El recurso se encuentra saturado.
- Los procesos que compiten por el recurso comienzan a interferirse unos a otros.
- Ej.: problema de la *hiperpaginación:*
  - Ocurre cuando el almacenamiento principal está lleno.
  - Los conjuntos de trabajo de los distintos procesos activos no pueden ser mantenidos simultáneamente en el almacenamiento principal.

Para *detectar los embotellamientos* se debe controlar cada *cola de peticiones* de los recursos; cuando una cola crece rápidamente significa que la *tasa de llegadas de peticiones* debe superar a su *tasa de servicio.*

El *aislamiento de los embotellamientos* es una parte importante de la *"afinación" de la "sintonización" del sistema.*

Los *embotellamientos* pueden *eliminarse:*

- Aumentando la capacidad de los recursos.
- Añadiendo más recursos de ése tipo en ése punto del sistema.

12.7 Ciclos de Retroalimentación

El rendimiento de un S. O. puede ser sensible al estado actual del sistema [7, Deitel].

Un *ciclo de retroalimentación* es una situación en la cual *la información del estado actual del sistema se pone a disposición de las peticiones entrantes.*

La *ruta de las peticiones* puede modificarse, si la retroalimentación indica que puede haber dificultad de darles servicio.

Retroalimentación negativa

La tasa de llegadas de nuevas peticiones puede decrecer como resultado de la información que se está retroalimentando.

Contribuye a la *estabilidad* de los sistemas de colas:

- Impide que las colas crezcan indefinidamente.
- Hace que la longitud de las colas se mantenga cerca de sus valores medios.

Retroalimentación positiva

La información retroalimentada provoca un *incremento* en vez de un decremento de algún parámetro.

Se deben *evitar* situaciones similares a la siguiente:

- El S. O. detecta capacidad disponible de cpu.
- El S. O. informa al planificador de trabajos que admita más trabajos en la mezcla de multiprogramación:
  - Con esto se incrementaría el uso de cpu.
- Al incrementarse la mezcla de multiprogramación:
  - Decrece la cantidad de memoria que se puede asignar a cada trabajo.
  - El número de fallos de página puede incrementarse.
  - La utilización de cpu puede decrecer.

Puede producir *inestabilidades:*

- Debe diseñarse con mucha prudencia.
- Se deben controlar los efectos de cada cambio incremental para ver si resulta una mejora anticipada.
- Si un cambio incremental deteriora el rendimiento se podría estar operando en un rango inestable.

Capítulo 13

Modelado Analítico en Relación al Rendimiento

13.1 Introducción al Modelado Analítico y Teoría de Colas

Algunas de las *técnicas* más conocidas de modelado analítico son [7, Deitel]:

- La teoría de colas.
- Los procesos de Markov.

Los *modelos analíticos:*

- Son las representaciones matemáticas de los sistemas.
- Permiten al evaluador del rendimiento sacar conclusiones acerca del comportamiento del sistema.

Las expresiones *teoría de colas* y *teoría de líneas (o filas) de espera* deben considerarse equivalentes [7, Deitel].

El término matemático *cola* significa una *línea de espera.*

Si no hubiera líneas de espera se podría recibir servicio de inmediato:

- Sería lo deseable.
- El costo de disponer de la suficiente capacidad de servicio para no tener que esperar sería muy elevado.

Se consume cierta cantidad de tiempo en líneas de espera por servicio pero:

- El costo de ese servicio es menor debido a la mejor utilización de la instalación de servicio.

Si existe una población de clientes que demandan cierto servicio prestado por servidores:

- Algunos clientes ingresarán a la red de colas y esperarán que un servidor quede disponible.

Algunas *colas* son:

- *Ilimitadas:* pueden crecer tanto como sea necesario para contener a los clientes que esperan.

- *Limitadas:* solo pueden contener un número fijo de clientes en espera y quizás hasta ninguno.

Se deben tener en cuenta *variables aleatorias* que pueden ser descritas por *distribuciones probabilísticas.*

La variable aleatoria *"q"* representa el *tiempo que emplea un cliente esperando en la cola a ser servido.*

La variable aleatoria *"s"* representa la cantidad de *tiempo que emplea un cliente en ser servido.*

La variable aleatoria *"w"* representa el *tiempo total que emplea un cliente en el sistema de colas: "w = q + s".*

13.2 Fuente, Llegadas y Llegadas de Poisson

Fuente

Los clientes son proporcionados a un sistema de colas desde una *fuente que puede ser infinita o finita [7, Deitel].*

Con una *fuente infinita* la cola de servicio puede llegar a ser *arbitrariamente grande.*

Para una *fuente finita* la cola de servicio es *limitada,* pero una fuente finita pero muy grande suele considerarse como infinita.

Llegadas

Supondremos que los clientes llegan a un *sistema de colas* en los *tiempos:*

$t_0 < t_1 < t_2 < \ldots < t_n$.

Los clientes llegan de uno en uno y nunca hay una colisión.

Las variables aleatorias "τ_k" miden los *tiempos entre las llegadas sucesivas* (arbitrario) y se denominan *tiempos entre llegadas:*

- Son variables aleatorias *independientes* y están *idénticamente distribuidas:*

  $\tau_k = t_k - t_{k-1}$, con *(k ≥ 1).*

Llegadas de Poisson

Las llegadas pueden seguir distintos patrones arbitrarios pero suele suponerse que forman un *proceso de llegadas de Poisson:*

- Los *tiempos entre llegadas* están distribuidos *exponencialmente:*

  $P(\tau \leq t) = 1 - e^{-\lambda t}$.

- La probabilidad de que lleguen *exactamente "n" clientes* en cualquier *intervalo de longitud "t"* es:

  $(e^{-\lambda t}(\lambda t)^n)/n!$, con *(n=0,1,2,...)*.

- λ es una *tasa promedio de llegadas constante* expresada en *"clientes por unidad de tiempo"*.

- El *número de llegadas por unidad de tiempo* se dice que tiene *distribución de Poisson con una media* λ .

13.3 Tiempos de Servicio, Capacidad de la Cola y Número de Servidores en el Sistema

Tiempos de servicio

Se supone que *los tiempos de servicio son aleatorios [7, Deitel]*.
"s_k" es el tiempo de servicio que el *k-ésimo cliente* requiere del sistema.
Un tiempo de servicio arbitrario se designa por *"s"*.
La *distribución de los tiempos de servicio* es:

$W_s(t) = P(s \leq t)$.

Para un *servicio aleatorio* con una *tasa promedio de servicio "μ":*

$W_s(t) = P(s \leq t) = 1 - e^{-\mu t}$, con $(t \geq 0)$.

Capacidad de la cola

Las colas deben tener:

- *Capacidad infinita:*
  - Cada cliente que llegue puede entrar en el sistema de colas y esperar, independientemente de cuántos clientes hay en espera [7, Deitel].

- *Capacidad cero (o sistemas de pérdidas):*
  - Los clientes que llegan cuando la instalación de servicio está ocupada no podrán ser admitidos al sistema.

- *Capacidad positiva:*
  - Los clientes que llegan solo esperan si hay lugar en la cola.

Número de servidores en el sistema

Los sistemas de colas se pueden categorizar según el número de servidores en:

- *Sistemas de un solo servidor:*

- Tienen un solo servidor y nada más pueden darle servicio a un solo cliente a la vez.

- *Sistemas de servidores múltiples:*

  - Tienen *"c"* servidores con idéntica capacidad y pueden dar servicio a *"c"* clientes a la vez.

13.4 Disciplinas de Colas

Son las reglas usadas para elegir al siguiente cliente de cola que va a ser servido [7, Deitel].

La más conocida es la *"FCFS"* o primero en llegar, primero en ser servido.

Generalmente se utilizan las siguientes *notaciones:*

- *Notación Kendall.*
- *Notación Kendall abreviada.*

Notación Kendall (A/B/c/K/m/Z):

- A: distribución de tiempos entre llegadas.
- B: distribución de tiempos de servicio.
- c: número de servidores.
- K: capacidad de cola del sistema.
- m: número de clientes en la fuente.
- Z: disciplina de cola.

Notación Kendall abreviada (A/B/c):

- No hay límite en la longitud de la cola.
- La fuente es infinita.
- La disciplina de cola es *"FCFS"*.
- "A" y "B" pueden ser:
  - GI: para tiempo entre llegadas *general independiente.*
  - G: para tiempo de servicio *general.*
  - E_k: para las distribuciones de tiempos entre llegadas o de servicio *Erlang-k.*
  - M: para las distribuciones de tiempos entre llegadas o de servicio *exponenciales.*
  - D: para las distribuciones de tiempos entre llegadas o de servicio *determinísticos.*
  - H_k: para las distribuciones de tiempos entre llegadas o de servicio *hiperexponenciales (con "k" estados).*

13.5 Intensidad de Tráfico y Utilización del Servidor

Intensidad de tráfico

Es una *medida de la capacidad del sistema* para dar servicio efectivo a sus clientes [7, Deitel].

Se define como la razón de la *media del tiempo de servicio "E(s)"* y la *media del tiempo entre llegadas "E(τ)".*

La *intensidad de tráfico "u"* es:

$u = [E(s)]/[E(\tau)] = \lambda E(s) = (\lambda/\mu)$:

λ: *tasa de llegadas.*

μ: *tasa de servicio.*

Es útil para *determinar el número mínimo de servidores idénticos* que necesitará un sistema para dar servicio a sus clientes:

- Sin que las colas se hagan indefinidamente largas.
- Sin tener que rechazar clientes.
- Ej.: si *E(s)* = 17 segundos y *E(τ)* = 5 segundos, u = 17 / 5 = 3,4:
  - El sistema deberá tener un mínimo de 4 servidores.

Se debe tener en cuenta que:

- La *tasa de llegadas* de los clientes es un *promedio:*
  - Es posible que no llegue ningún cliente durante un largo tiempo.
  - Es posible que los clientes lleguen en rápida sucesión, excediendo la capacidad física de la cola y ocasionando el rechazo de clientes.
  - Si se utilizan colas de tamaño fijo debe haber capacidad suficiente para soportar excesos ocasionales de la tasa de llegadas.
  - Se podrían utilizar colas de longitud variable junto con lista encadenada.

Utilización del servidor

Se define como la *intensidad de tráfico por servidor:*

$\rho = u/c = \lambda/\mu c.$

Es la *probabilidad de que un servidor determinado se encuentre ocupado.*

Según la *ley de los grandes números* esta probabilidad es aproximadamente *la fracción de tiempo que cada servidor está en uso.*

En sistemas de *un solo servidor* es igual a la *intensidad de tráfico.*

13.6 Estado Estable en Función de Soluciones Transitorias

Los sistemas de colas que se han *"asentado"* se dice que están operando en *estado estable [7, Deitel].*

Generalmente la operación inicial de un sistema no es indicativa de su comportamiento periódico.

Los sistemas de colas deben pasar por algún período inicial de operación antes de tener un funcionamiento:

- Uniforme.
- Predecible.

La solución y estudio de un sistema de colas se simplifica mucho si se sabe que se encuentra en *estado estable:*

- Ciertos parámetros importantes permanecen fijos.
- Resulta relativamente fácil categorizar la operación del sistema.

Las *soluciones transitorias* o *dependientes del tiempo:*

- Son mucho más complejas.
- Están fuera del alcance de este curso.

13.7 Resultado de Little

Es una de las mediciones más sencillas y útiles del rendimiento de un sistema de colas [7, Deitel].

Relaciona las siguientes cantidades:

- W_q: *tiempo medio* que emplea un *cliente en una cola.*
- λ: *tasa de llegadas.*
- L_q: *número* de *clientes en la cola.*
- W: *tiempo medio* que emplea un *cliente en el sistema.*
- L: *número* de *clientes en el sistema.*

El *resultado de Little* se expresa como:

$L_q = \lambda W_q$

$L = \lambda W$

13.8 Resumen del Proceso de Poisson

Se define *"P(k,t)" como la probabilidad de exactamente "k" llegadas en un intervalo de tiempo de longitud "t" [7, Deitel].*

Un *proceso es de Poisson* si y solo si:

- Para intervalos apropiadamente pequeños Δt:

  P(k,t) =

  $\lambda \Delta t$ para k = 1 (λ es la *tasa promedio de llegadas*).
  $1 - \lambda \Delta t$ para k = 0.
  0 para k > 1.

- Cualesquiera *eventos* definidos para tener lugar en intervalos de tiempo no superpuestos son *mutuamente independientes.*

Un *proceso también es de Poisson* si los tiempos entre llegadas sucesivas *(tiempos entre llegadas de primer orden):*

- Son variables aleatorias exponenciales.
- Idénticamente distribuidas.

Si la variable aleatoria *"k"* indica el número de llegadas:

- La *probabilidad* de, exactamente, *"k" llegadas en un intervalo de longitud "t"* es:

  $P(k,t) = [(\lambda t)^k e^{-\lambda t}]/k!; \qquad t \geq 0; \qquad k = 0, 1, 2, \ldots$

- El *valor esperado* o *valor medio* de *"k"* es:

  E(k) = λ t.

- La *varianza* de *"k"* es:

  $(\sigma_k)^2 = \lambda t.$

La *suma de dos variables de Poisson* aleatorias independientes *"x"* e *"y"* también describen un *proceso de Poisson:*

- Los *valores esperados* son:

  $E(y) = \mu_2 = \lambda_2 t.$
  $E(x) = \mu_1 = \lambda_1 t.$

- La *probabilidad de "k" llegadas en el tiempo "t"* es:

  $P(k,t) = [(\lambda_1 + \lambda_2)^k e^{-(\lambda_1 t + \lambda_2 t)}]/k!; \qquad t \geq 0; \qquad k = 0, 1, 2, ...$

$P(k,t) = [(\mu_1 + \mu_2)^k e^{-(\mu_1+\mu_2)}]/k!.$
$P(k,t) = [(\mu_s{}^k e^{-\mu_s}]/k!; \qquad \mu_s = \mu_1 + \mu_2.$
$P(k,t) = [(\lambda_s t)^k e^{-\lambda_s t}]/k!; \qquad \lambda_s = \lambda_1 + \lambda_2$

La suma de *"n" procesos de Poisson independientes* resulta en un *proceso de Poisson* con una *tasa de llegada:*

$\lambda = \sum_{i=1}^{n} \lambda_i$

Para un *proceso de Poisson con una tasa de llegada "λ"* se puede formar un *nuevo proceso de Poisson* utilizando *borradas aleatorias independientes:*

- Cada llegada al proceso original:
  - Se *acepta al nuevo proceso* con probabilidad *"P".*
  - Se *rechaza* con probabilidad *"1 - P".*
- La *tasa de llegada del nuevo proceso* derivado es "λP" [7, Deitel].

La generalización para la *descomposición de un proceso de Poisson en "n" procesos derivados independientes*, cada uno con una *probabilidad asociada "p_i"* resulta:

$\lambda_n = p_n \lambda.$

$\sum_{i=1}^{n} p_i = 1.$

$\sum_{i=1}^{n} \lambda_i = \sum_{i=1}^{n} p_i \lambda = \lambda \sum_{i=1}^{n} p_i = \lambda.$

En un *proceso de Poisson:*

- La *probabilidad de que no haya llegadas en un intervalo de longitud "t"* es:

  $P(0,t) = [(\lambda t)^0 e^{-\lambda t}]/0! = e^{-\lambda t}.$

- La *probabilidad de una o más llegadas en un intervalo de longitud "t"* es:

  $1 - P(0,t) = 1 - e^{-\lambda t}.$

La *función de densidad de probabilidad* para el *tiempo entre llegadas de primer orden (tiempo hasta la primer llegada)* es:

$f_t(t) = \lambda e^{-\lambda t}; \qquad (t \geq 0).$

El *valor esperado "t"* es:

$E(t) = 1/\lambda.$

La *varianza* es:

$\sigma_t^2 = 1/\lambda^2$.

La *función de densidad de probabilidad* para el *tiempo entre llegadas de orden r-ésimo (tiempo hasta la r-ésima llegada)* es:

$f_t(t) = (\lambda^r t^{r-1} e^{-\lambda t})/(r-1)!; \qquad (t \geq 0, r = 1, 2, ...)$.

El *valor esperado "t"* es:

$E(t) = r/\lambda$.

La *desviación estándar* es:

$\sigma_t^2 = r/\lambda^2$.

Las instalaciones de servicio pueden proporcionar *tiempos de servicio exponenciales:*

- La *probabilidad de que el tiempo de servicio sea menor o igual a "t"* es:

  $P(S \leq t) = 1 - e^{-\mu t}; \qquad (t \geq 0)$.

- La *tasa promedio de servicio es "μ".*
- El *tiempo promedio de servicio es "1 / μ".*
- La *función de densidad de probabilidad para el tiempo de servicio "t"* es:

  $f_t(t) = \mu e^{-\mu t}; \qquad (t \geq 0)$.

- La *media del tiempo de servicio* es:

  $E(s) = 1/\mu$.

- La *varianza* es "$1/\mu^2$".

Un servidor que opera de esta manera se denomina *servidor exponencial.*

13.9 Análisis de un Sistema de Colas M/M/1

Las *fórmulas de estado para el sistema de colas M/M/c* son las siguientes [7, Deitel]*:*

- *Intensidad de tráfico:*

  $u = \lambda/\mu = \lambda E(s)$.

- *Utilización del servidor:*

  $\rho = u/c$.

- *Probabilidad de que todos los servidores estén en uso*, por lo que un cliente que llega debe esperar:

  $C(c,u) = \left\{ \frac{[(u^c)/c!]}{[(u^c)/c!]+(1-\rho)\left[\sum_{n=0}^{c-1}[u^n/n!]\right]} \right\}$

- *Tiempo promedio en la cola:*

  $W_q = \frac{C(c,u)E(s)}{c(1-\rho)}$

- *Tiempo promedio en el sistema:*

  $W = W_q + E(s).$

- *Percentil 90 de tiempo de espera en la cola:*

  $\pi_q(90) = \{[E(s)]/[c(c-\rho)]\}\{ln[10C(c,u)]\}.$

Las *fórmulas de estado para el sistema de colas M/M/1* son las siguientes:

- Se deducen de las anteriores:

  $C(c,u) = \rho = \lambda E(s).$
  $W_q = [\rho E(s)]/(1-\rho).$
  $W = E(s)/(1-\rho).$
  $\pi_q(90) = W[ln(10\rho)].$

Seguidamente se detalla un *ejemplo* para el análisis:

- Los operadores de una empresa precisan usar un equipo especial.
- La empresa opera las 24 hs. del día.
- Los 48 operadores (como promedio) necesitan usar el equipo una vez al día.
- Los operadores llegan al equipo en forma aleatoria (llegadas de Poisson).
- El tiempo que cada operador utiliza el equipo es exponencial y como promedio es de 20 minutos.

Utilizando un *sistema de colas m/m/1* para modelar el uso del equipo especial del ejemplo se obtiene:

- *Utilización del equipo:*

  $u = \lambda E(s) = (48/24).(1/3) = 2/3; \rho = 2/3;$ *E(s)* = 20 minutos.

- *Tiempo promedio de espera de un operador antes de usar el equipo:*

  $W_q = [\rho E(s)]/(1-\rho) =$ [(2 / 3). 20] / (1 /3) = 40 minutos.

- *Tiempo total que un operador utiliza el equipo:*

  $W = W_q + E(s) = 40$ min. + 20 min. = 60 minutos.

- *Percentil 90 de tiempo de espera en la cola:*

  $\pi_q(90) = W[ln(10\rho] = 60$ ln (6,667) = 113,826 minutos:

  - ∗ Un 10 % de los operadores (unos 5 por día) sufre prolongadas esperas de casi 2 horas.

Según el *resultado de Little:*

- *Tasa de llegada de operadores al equipo:*

  $\lambda = 48$ / 24 (60) = 1 / 30 operadores por minuto.

- *Operadores en espera:*

  $L_q = (1 / 30) . 40 = 1{,}33$ operadores en espera.

- *Operadores en el sistema:*

  $L = (1 / 30) . 60 = 2$ operadores en el cuarto del equipo.

Conclusión:

- *Un solo equipo no es suficiente para hacer frente a las necesidades de los operadores sin provocar esperas excesivas.*

13.10 Análisis de un Sistema de Colas M/M/c

Seguidamente se detalla un *ejemplo* para el análisis, que es el mismo del tema anterior [7, Deitel]:

- Si la decisión es adquirir más equipos, los interrogantes son los siguientes:
  - ¿Cuántos equipos adicionales deberán adquirirse para mantener el percentil 90 de tiempo en espera por debajo de 10 minutos?.
  - ¿Deberán mantenerse todos los equipos en un lugar central, o deberán distribuirse por todo el edificio?:
    - ∗ (*Nota:* se debe ignorar el tiempo que les lleva a los operadores llegar hasta los equipos).

Colocando los equipos en diferentes lugares de la empresa:

- Cada uno de los equipos se debe analizar como un *sistema de colas M/M/1.*

- La carga de trabajo debe dividirse en partes iguales entre los equipos.
- Colocando 1, 2, 3, 4 o 5 equipos en localidades separadas obtenemos los siguientes valores:
  - Utilización del servidor:ρ: 2/3; 1/3; 2/9; 1/6 y 2/15.
  - Tiempo de espera de servicio: *E(s)*: 20 minutos en todos los casos.
  - Tiempo de espera en la cola: W_q: 40; 10; 5,7; 4 y 3,1 minutos.
  - Tiempo de espera en el sistema: W: 60; 30; 25,7; 24 y 23 minutos.
  - Percentil 90 de tiempo de espera en la cola:$\pi_q(90)$: 113,8; 36,1; 20,5; 12,3 y 6,6 minutos.
- *Conclusiones:*
  - Los tiempos de espera en la cola bajan muy rápido tan pronto como se añade el segundo equipo M/M/1.
  - El percentil 90 de tiempo de espera en la cola es superior a 10 minutos hasta que se añade el quinto equipo.

Colocando todos los equipos en un lugar central:

- Se considera un *sistema de colas M/M/2 sencillo.*
- Utilizando las fórmulas de los *sistemas M/M/c* se obtienen los siguientes valores:
  - Intensidad de tráfico: *u:* 2/3.
  - Utilización del servidor: ρ : 1/3.
  - Probabilidad de que todos los servidores se encuentren en este momento en uso, por lo que un cliente que llega debe esperar: *C(c,u):* 1/6.
  - Tiempo promedio en la cola: W_q: 2,5 minutos.
  - Tiempo promedio en el sistema: *W:* 22,5 minutos.
  - Percentil 90 de tiempo de espera en la cola: $\pi_q(90)$: 7,66 minutos.
- *Conclusiones:*
  - El percentil 90 de tiempo de espera en la cola del sistema M/M/2 es inferior al criterio de 10 minutos.
  - Con solo 2 equipos centralizados en una posición se pueden eliminar los problemas de espera del sistema de un solo equipo.
  - Para asegurar un percentil 90 de tiempo de espera en la cola inferior a 10 minutos serán necesarios:
    - 5 equipos M/M/1 distribuidos, o.
    - 2 equipos en una configuración M/M/2 central.

13.11 Procesos de Markov

Un *proceso de Markov es un modelo adecuado* para describir el comportamiento de sistemas donde el sistema está situado en uno de un *conjunto de estados discretos mutuamente excluyentes y colectivamente exhaustivos* S_0, S_1, S_2,...., S_n [7, Deitel].

El *estado presente del sistema* y las *probabilidades de transición* entre varios estados del sistema, caracterizan el *comportamiento futuro del sistema.*

Dado que un proceso de Markov se encuentra en un *estado determinado*, su comportamiento futuro *no depende de su historia anterior* a su entrada a ese estado.

Muchos procesos de Markov exhiben un *comportamiento de estado estable*, es decir que las probabilidades de que el proceso se encuentre en un estado determinado son *constantes en el tiempo.*

Se dice que un *estado "S_j" es transitorio* si desde un estado *"S_k"* que puede ser alcanzado desde *"S_j"*, el sistema no puede regresar a *"S_j"*.

Se dice que un *estado "S_j" es recurrente* si desde cada estado *"S_k"* alcanzable desde *"S_j"*, el sistema puede regresar a *"S_k"*.

Una *cadena sencilla* es una serie de estados recurrentes tal que el sistema puede llegar a cualquier estado de la cadena desde cualquier otro estado de esta.

Un *cambio de estado* en un *proceso de Markov de transición continua* puede producir cambios de estado en cualquier instante de una escala de tiempo continua.

13.12 Procesos de Nacimiento y Muerte

Son un caso importante de los *procesos de Markov [7, Deitel].*

Son particularmente aplicables al *modelado* de sistemas de computación.

Un *proceso de Markov de nacimiento y muerte continuo* tiene la propiedad de que:

- $\lambda_{ij} = 0$ si $j \neq i + 1$ y $j \neq i - 1$.
- λ_{ij} es la tasa a la cual ocurren las *transiciones* del estado *"S_i"* al estado *"S_j"*.
- $\lambda_{i(i+1)} = b_i$ es la *tasa promedio de nacimiento* desde el estado *"S_i"*.
- $\lambda_{i(i-1)} = d_i$ es la *tasa promedio de muerte* desde el estado *"S_i"*.
- *"P_i"* es la *probabilidad de estado estable* de que el proceso se encuentre en el estado *"S_i"*.

En estado estable, en cualquier intervalo de tiempo aleatorio *"Δt"*, el proceso puede realizar las siguientes *transiciones* con la misma probabilidad:

$S_i \rightarrow S_{i+1}$ con una probabilidad $P_i b_i$.

$S_{i+1} \rightarrow S_i$ con una probabilidad $P_{i+1} d_{i+1}$.

$P_i b_i = P_{i+1} d_{i+1}$.

La *resolución de un proceso de nacimiento y muerte continuo* significa determinar los diferentes *"P_i"* usando las relaciones:

$P_{i+1} = (b_i \ / \ d_{i+1}) \ P_i$.

$\sum_{i=1}^{n} P_i = 1$.

13.13 Análisis del Rendimiento de un Subsistema de Disco

Se supone la siguiente situación [7, Deitel]:

- Las *peticiones de acceso a disco* llegan como un *proceso de Poisson* con una *tasa promedio de "λ" peticiones por minuto.*
- Si el disco está en uso, la petición se coloca en una *cola primero en llegar, primero en ser servido.*
- Cuando el disco queda disponible se sirve la primera petición de la cola.
- El *tiempo de servicio* es una variable aleatoria exponencialmente distribuida con un valor *esperado de "1 / μ" minutos.*
- La *tasa promedio de servicio es de "μ" peticiones por minuto.*

Se debe *determinar*, para cada uno de los casos:

- El *valor esperado* para el número total de peticiones al disco pendientes (en la cola o en servicio).
- Las *probabilidades* del estado límite.

Caso I:

- El dispositivo de disco contiene un solo brazo.
- Solo puede dar servicio a una petición a la vez.
- La *tasa de servicio* es "μ".

Caso II:

- El dispositivo de disco contiene gran número de brazos móviles.
- Cada brazo puede dar servicio a una petición de disco a la misma tasa "μ".
- Se supone que un número infinito de peticiones pueden recibir servicio en paralelo.

Solución al caso I

"S_i" es el estado del sistema cuando hay *"i" peticiones* de disco al dispositivo de servicio de disco.

La *tasa de llegadas* de peticiones es independiente del estado del sistema:

- La *probabilidad de la transición "S_i" $\rightarrow$ "S_{i+1}"* en el siguiente intervalo de tiempo *"Δt"* es *"$\lambda \Delta t$".*

Se considera al sistema como un *proceso de nacimiento y muerte continuo de cadena sencilla y estados infinitos con:*

- $d_i = 0$ con i = 0.
- $di = \mu$ con i = 1, 2, 3, ...
- $bi = \lambda$ con i = 0, 1, 2, ...
- Solo una petición puede ser servida en un momento dado y se sirve a una tasa μ.
- $\mu > \lambda$:
  - Asegura que la longitud de la cola de peticiones en espera *no crezca indefinidamente.*

Se utilizan las relaciones:

$P_{i+1} = (b_i / d_{i+1}) P_i$; i = 0, 1, 2, ...

$\sum_i P_i = 1$.

$P_1 = (\lambda / \mu) P_0$.

$P_2 = (\lambda / \mu) P_1 = (\lambda / \mu)^2 P_0$.

$P_i = (\lambda / \mu)^i P_0$.

$\sum_i P_i = 1 = \sum_i (\lambda / \mu)^i P_0 = 1/ [1 - (\lambda / \mu)] P_0$.

$\mathbf{P}_0 = 1 - (\lambda / \mu)$: *probabilidad de que el sistema se encuentre ocioso.*

$P_i = (\lambda / \mu)^i P_0 = [1 - (\lambda / \mu)] (\lambda / \mu)^i$. i = 0, 1, 2, ...

$\mathbf{P}_i = [1 - (\lambda / \mu)] (\lambda / \mu)^i$: *probabilidad que hayan "i" peticiones pendientes.*

El *número promedio de peticiones pendientes* es:

$E(i) = \sum_i i P_i = \sum_i i [1 - (\lambda / \mu)] (\lambda / \mu)^i = [1 - (\lambda / \mu)] \sum_i i (\lambda / \mu)^i =$

$E(i) = [1 - (\lambda / \mu)] (\lambda / \mu) \sum_i i (\lambda / \mu)^{i-1} =$

$E(i) = [1 - (\lambda / \mu)] (\lambda / \mu) \{1 /[1 - (\lambda / \mu)^2]\} =$

$\mathbf{E(i)} = (\lambda / \mu) [1 - (\lambda / \mu)^{-1}]$.

Solución al caso II
Con *"i" peticiones* siendo servidas:

- La *probabilidad de que una petición en particular acabe siendo servida* dentro del siguiente *"Δt"* es *"$\mu\Delta t$"*.

- La *probabilidad de que exactamente una petición cualquiera acabe* es *"i$\mu\Delta t$"* (buena aproximación de primer orden).
- Cualquiera de las *"i" peticiones* puede terminar y provocar un *cambio de estado.*

El sistema se ve como un *proceso de nacimiento y muerte continuo de cadena sencilla y de estados infinitos con:*

- $b_i = \lambda$ con i = 0, 1, 2, ...
- $d_i = 0$ con i = 0.
- $d_i = i\mu$ con i = 1, 2, 3, ...

Ningún cliente tiene que esperar ya que *se suponen infinitos servidores en paralelo.*
Se utilizan las relaciones:

$P_{i+1} = (b_i \ / \ d_{i+1}) \ P_i;$ i = 0, 1, 2, ...

$\sum_i P_i = 1.$

$P_1 = (\lambda \ / \ \mu) \ P_0.$

$P_2 = (\lambda \ / \ 2\mu) \ P_1 = (1 \ / \ 2) \ (\lambda \ / \ \mu)^2 \ P_0.$

$P_3 = (\lambda \ / \ 3\mu) \ P_2 = (1 \ / \ (3 \ . \ 2)) \ (\lambda \ / \ \mu)^3 \ P_0.$

$P_i = (1 \ / \ i!) \ (\lambda \ / \ \mu)^i \ P_0.$

$\sum_i P_i = 1 = \sum_i (1 \ / \ i!) \ (\lambda \ / \ \mu)^i \ P_0.$

$\sum_n (x^n \ / \ n!) = e^x.$

$\sum_i P_i = 1 = \sum_i (1 \ / \ i!) \ (\lambda \ / \ \mu)^i \ P_0 = e^{\lambda/\mu} \ P_0.$

$\mathbf{P}_0 = e^{-\lambda/\mu}$.

$\mathbf{P}_i = (\lambda \ / \ \mu)^i \ [(e^{-\lambda/\mu}) \ / \ i!].$

$E(i) = \sum_i iP_i = \sum_i i \ (\lambda \ / \ \mu)^i \ [(e^{-\lambda/\mu}) \ / \ i!] = (e^{-\lambda/\mu}) \sum_i i \ (\lambda \ / \ \mu)^i \ (1 \ / \ i!) =$

$E(i) = (e^{-\lambda/\mu}) \sum_i (\lambda \ / \ \mu) \ (\lambda \ / \ \mu)^{i-1} \ [1 \ / \ (i - 1)!] =$

$E(i) = (e^{-\lambda/\mu}) \ (\lambda \ / \ \mu) \sum_i [1 \ / \ (i - 1)!] \ (\lambda \ / \ \mu)^{i-1} =$

$E(i) = (e^{-\lambda/\mu}) \ (\lambda \ / \ \mu) \ (e^{\lambda/\mu}) =$

$\mathbf{E(i)} = (\lambda \ / \ \mu).$

Conclusiones:

- En el sistema de *un solo servidor*, si una petición que llega encuentra ocupado el dispositivo de disco, debe esperar.

- En el sistema de *servidores infinitos*, las peticiones que llegan siempre entran al servicio de inmediato.
- En el sistema de *un solo servidor:*
  - A medida que λ tiende a μ la probabilidad de que el sistema se encuentre ocioso decrece rápidamente:
    - Las peticiones que llegan esperan.
  - El número promedio de peticiones pendientes crece con rapidez.
- En el sistema de *servidores infinitos:*
  - El número promedio de peticiones pendientes tiende a 1.

Capítulo 14

Seguridad de los Sistemas Operativos

14.1 Introducción a la Seguridad de los Sistemas Operativos

La evolución de la computación y de las comunicaciones en las últimas décadas [7, Deitel]:

- Ha hecho más accesibles a los sistemas informáticos.
- Ha incrementado los riesgos vinculados a la seguridad.

La *vulnerabilidad de las comunicaciones de datos* es un aspecto clave de la *seguridad* de los sistemas informáticos; la importancia de este aspecto es cada vez mayor en función de la proliferación de las *redes de computadoras.*

El *nivel de criticidad y de confidencialidad de los datos* administrados por los sistemas informáticos es cada vez mayor:

- Ej.: correo personal, transferencia de fondos, control de manufactura, control de sistemas de armas, control de tráfico aéreo, control de implantes médicos (marcapasos, etc.).
- Los sistemas deben funcionar ininterrumpidamente y sin problemas.

El *sistema operativo*, como *administrador de los recursos* del sistema:

- Cumple una función muy importante en la instrumentación de la seguridad.
- No engloba a todos los aspectos de la seguridad.
- Debe ser complementado con medidas externas al S. O.

La simple *seguridad física resulta insuficiente* ante la posibilidad de acceso mediante equipos remotos conectados.

La *tendencia* es que los sistemas sean *más asequibles y fáciles de usar*, pero la *favorabilidad* hacia el usuario puede implicar un *aumento de la vulnerabilidad.*

Se deben *identificar las amenazas potenciales*, que pueden proceder de fuentes maliciosas o no.

El nivel de seguridad a proporcionar depende del valor de los recursos que hay que asegurar.

14.2 Requisitos de Seguridad

Los *requisitos de seguridad* de un sistema dado definen *lo que significa la seguridad*, para ese sistema [7, Deitel].

Los requisitos sirven de base para *determinar si el sistema implementado es seguro:*

- Sin una serie de requisitos precisos tiene poco sentido cuestionar la seguridad de un sistema.
- Si los requisitos están débilmente establecidos no dicen mucho sobre la verdadera seguridad del sistema.

Algunos *ejemplos de formulación* de los requisitos de seguridad son los siguientes:

- *Directiva DOD 5200.28 (EE. UU.):*
  - Especifica cómo debe manipularse la información clasificada en sistemas de procesamiento de datos.
- *Manual de Referencia de Tecnología de Seguridad de la Computadora (EE. UU.):*
  - Especifica cómo evaluar la seguridad de los sistemas de computación de la Fuerza Aérea.
- *Ley de Intimidad de 1974 (EE. UU.):*
  - Requiere que las Agencias Federales aseguren la integridad y seguridad de la información acerca de los individuos, especialmente en el contexto del amplio uso de las computadoras.

14.3 Un Tratamiento Total de la Seguridad

Un *tratamiento total* incluye aspectos de la seguridad del computador *distintos a los de la seguridad de los S. O. [7, Deitel].*

La *seguridad externa* debe asegurar la instalación computacional contra intrusos y desastres como incendios e inundaciones:

- Concedido el acceso físico el S. O. debe identificar al usuario antes de permitirle el acceso a los recursos: *seguridad de la interfaz del usuario.*

La *seguridad interna* trata de los controles incorporados al hardware y al S. O. para asegurar la *confiabilidad, operabilidad* y la *integridad* de los programas y datos.

14.4 Seguridad Externa y Seguridad Operacional

14.4.1 Seguridad Externa

La *seguridad externa* consiste en [7, Deitel]:

- Seguridad física.
- Seguridad operacional.

La *seguridad física* incluye:

- Protección contra desastres.
- Protección contra intrusos.

En la seguridad física son importantes los *mecanismos de detección*, algunos ejemplos son:

- Detectores de humo.
- Sensores de calor.
- Detectores de movimiento.

La *protección contra desastres* puede ser costosa y frecuentemente no se analiza en detalle; depende en gran medida de las consecuencias de la pérdida.

La *seguridad física* trata especialmente de impedir la entrada de intrusos:

- Se utilizan sistemas de *identificación física:*
  - Tarjetas de identificación.
  - Sistemas de huellas digitales.
  - Identificación por medio de la voz.

14.4.2 Seguridad Operacional

Consiste en las *diferentes políticas y procedimientos* implementados por la administración de la instalación computacional [7, Deitel].

La *autorización* determina qué acceso se permite y a quién.

La *clasificación* divide el problema en subproblemas:

- Los datos del sistema y los usuarios se dividen en *clases:*
  - A las clases se conceden diferentes *derechos de acceso.*

Un aspecto *crítico* es la *selección y asignación de personal:*

- La pregunta es si se puede *confiar en la gente.*

- El tratamiento que generalmente se da al problema es la *división de responsabilidades:*
  - Se otorgan distintos conjuntos de responsabilidades.
  - No es necesario que se conozca la totalidad del sistema para cumplir con esas responsabilidades.
  - Para poder comprometer al sistema puede ser necesaria la cooperación entre muchas personas:
    - Se reduce la probabilidad de violar la seguridad.
  - Debe instrumentarse un gran número de verificaciones y balances en el sistema para ayudar a la detección de brechas en la seguridad.
  - El personal debe estar al tanto de que el sistema dispone de controles, pero:
    - Debe desconocer cuáles son esos controles:
      - Se reduce la probabilidad de poder evitarlos.
    - Debe producirse un efecto disuasivo respecto de posibles intentos de violar la seguridad.

Para diseñar *medidas efectivas de seguridad* se debe primero:

- Enumerar y comprender las amenazas potenciales.
- Definir qué grado de seguridad se desea (y cuánto se está dispuesto a gastar en seguridad).
- Analizar las contramedidas disponibles.

14.5 Vigilancia, Verificación de Amenazas y Amplificación

14.5.1 Vigilancia

La *vigilancia* tiene que ver con [7, Deitel]:

- La verificación y la auditoría del sistema.
- La autentificación de los usuarios.

Los sistemas sofisticados de *autentificación de usuarios* resultan muy difíciles de evitar por parte de los intrusos.

Un problema existentes es la posibilidad de que el sistema *rechace* a usuarios legítimos:

- Un sistema de reconocimiento de voz podría rechazar a un usuario legítimo resfriado.
- Un sistema de huellas digitales podría rechazar a un usuario legítimo que tenga una cortadura o una quemadura.

14.5.2 Verificación de Amenazas

Es una técnica según la cual los usuarios *no pueden tener acceso directo a un recurso* [7, Deitel]:

- Solo lo tienen las rutinas del S. O. llamadas *programas de vigilancia.*
- El usuario solicita el acceso al S. O.
- El S. O. niega o permite el acceso.
- El acceso lo hace un programa de vigilancia que luego pasa los resultados al programa del usuario.
- Permite:
  - Detectar los intentos de penetración en el momento en que se producen.
  - Advertir en consecuencia.

14.5.3 Amplificación

La *amplificación* se produce cuando [7, Deitel]:

- Un *programa de vigilancia* necesita para cumplir su cometido mayores derechos de acceso de los que disponen los usuarios:
  - Ej.: se requiere calcular un promedio para lo cual es necesario leer un conjunto de registros a los que el usuario no tiene acceso individualmente.

14.6 Protección por Contraseña

Las clases de elementos de *autentificación* para establecer la *identidad de una persona* son [7, Deitel]:

- Algo *sobre la persona:*
  - Ej.: huellas digitales, registro de la voz, fotografía, firma, etc.
- Algo *poseído por la persona:*
  - Ej.: insignias especiales, tarjetas de identificación, llaves, etc.
- Algo *conocido por la persona:*
  - Ej.: contraseñas, combinaciones de cerraduras, etc.

El esquema más común de autentificación es la *protección por contraseña:*

- El usuario elige una *palabra clave*, la memoriza, la teclea para ser admitido en el sistema computarizado:

  - La clave no debe desplegarse en pantalla ni aparecer impresa.

La protección por contraseñas tiene ciertas *desventajas* si no se utilizan criterios adecuados para:

- Elegir las contraseñas.
- Comunicarlas fehacientemente en caso de que sea necesario.
- Destruir las contraseñas luego de que han sido comunicadas.
- Modificarlas luego de algún tiempo.

Los *usuarios* tienden a elegir contraseñas *fáciles de recordar:*

- Nombre de un amigo, pariente, perro, gato, etc.
- Número de documento, domicilio, patente del auto, etc.

Estos datos *podrían ser conocidos* por quien intente una violación a la seguridad mediante intentos repetidos, por lo tanto debe *limitarse la cantidad de intentos fallidos* de acierto para el ingreso de la contraseña.

La contraseña no debe ser muy corta para no facilitar la probabilidad de acierto.

Tampoco debe ser muy larga para que no se dificulte su memorización, ya que los usuarios la anotarían por miedo a no recordarla y ello incrementaría los riesgos de que trascienda.

14.7 Auditoría y Controles de Acceso

14.7.1 Auditoría

La auditoría suele realizarse *a posteriori* en *sistemas manuales* [7, Deitel], es decir que se examinan las recientes transacciones de una organización para determinar si hubo ilícitos.

La auditoría en un *sistema informático* puede implicar un *procesamiento inmediato*, pues se verifican las transacciones que se acaban de producir.

Un *registro de auditoría* es un registro permanente de acontecimientos importantes acaecidos en el sistema informático:

- Se realiza automáticamente cada vez que ocurre tal evento.
- Se almacena en un área altamente protegida del sistema.
- Es un mecanismo importante de detección.

El registro de auditoría debe ser *revisado* cuidadosamente y con frecuencia:

- Las revisiones deben hacerse:
  - Periódicamente:
    - Se presta atención regularmente a los problemas de seguridad.
  - Al azar:
    - Se intenta atrapar a los intrusos desprevenidos.

14.7.2 Controles de Acceso

Lo fundamental para la *seguridad interna* es *controlar el acceso a los datos almacenados [7, Deitel].*

Los *derechos de acceso* definen *qué acceso* tienen varios sujetos o varios objetos.

Los sujetos acceden a los objetos.

Los *objetos* son entidades que contienen *información.*

Los *objetos* pueden ser:

- Concretos:
  - Ej.: discos, cintas, procesadores, almacenamiento, etc.
- Abstractos:
  - Ej.: estructuras de datos, de procesos, etc.

Los objetos están *protegidos* contra los sujetos.

Las *autorizaciones* a un sistema se conceden *a los sujetos.*

Los *sujetos* pueden ser varios tipos de entidades:

- Ej.: usuarios, procesos, programas, otras entidades, etc.

Los *derechos de acceso* más comunes son:

- Acceso de lectura.
- Acceso de escritura.
- Acceso de ejecución.

Una forma de *implementación* es mediante una *matriz de control de acceso* con:

- Filas para los sujetos.
- Columnas para los objetos.
- Celdas de la matriz para los derechos de acceso que un usuario tiene a un objeto.

Una matriz de control de acceso debe ser muy celosamente protegida por el S. O.

14.8 Núcleos de Seguridad y Seguridad por Hardware

14.8.1 Núcleos de Seguridad

Es mucho más fácil hacer un sistema más seguro si la seguridad se ha incorporado desde el principio al diseño del sistema [7, Deitel].

Las *medidas de seguridad* deben ser implementadas en *todo el sistema informático.*

Un sistema de alta seguridad requiere que el *núcleo del S. O.* sea seguro.

Las medidas de seguridad más decisivas se implementan en el *núcleo*, que se mantiene intencionalmente lo más pequeño posible.

Generalmente se da que aislando las funciones que deben ser aseguradas en un S. O. de propósito general a gran escala, se crea un núcleo grande.

La *seguridad del sistema* depende especialmente de *asegurar las funciones que realizan:*

- El control de acceso.
- La entrada al sistema.
- La verificación.
- La administración del almacenamiento real, del almacenamiento virtual y del sistema de archivos.

14.8.2 Seguridad por Hardware

Existe una tendencia a *incorporar al hardware funciones del S. O. [7, Deitel]:*

- Las *funciones* incorporadas al hardware:
  - Resultan mucho *más seguras* que cuando son asequibles como instrucciones de software que pueden ser modificadas.
  - Pueden operar mucho *más rápido* que en el software:
    * Mejorando la performance.
    * Permitiendo controles más frecuentes.

14.9 Sistemas Supervivientes

El diseño de *sistemas de alta seguridad* debe asegurar [7, Deitel]:

- Su operación de manera continua y confiable.
- Su disponibilidad.

Un *sistema de computación superviviente* es aquel que *continúa operando aún después de que uno o más de sus componentes falla:*

- Es una cuestión cada vez más importante, especialmente para *sistemas en línea.*

Generalmente continúan operando con una *degradación suave* en los niveles de prestación.

Los componentes fallidos deben poder *reemplazarse sin interrumpir* el funcionamiento del sistema.

Una clave para la capacidad de supervivencia es la *redundancia:*

- Si un componente falla, otro equivalente toma su puesto.

- Se puede implementar como:
  - Un conjunto de recursos idénticos que funcionan en paralelo.
  - Un conjunto separado de recursos redundantes que se activan cuando se produce un fallo.

Algunas *características de supervivencia* son:

- La incorporación de *mecanismos contra fallos en el hardware* en vez de en el software.
- El uso de *multiprocesamiento transparente* para permitir mejorar el rendimiento sin modificar el software.
- El uso de *subsistemas múltiples* de entrada / salida.
- La incorporación de *mecanismos de detección de fallos* en el hardware y en el software.

14.10 Capacidades y Sistemas Orientados Hacia el Objeto

Un *derecho de acceso* permite a algún *sujeto* acceder a algún *objeto* de una *manera preestablecida [7, Deitel].*

Los *sujetos son los usuarios* de los sistemas de computación o entidades que *actúan en nombre:*

- De los usuarios.
- Del sistema.
- Ej.: trabajos, procesos y procedimientos, etc.

Los *objetos son los recursos* del sistema:

- Ej.: archivos, programas, semáforos, directorios, terminales, canales, dispositivos, pistas de discos, bloques de almacenamiento primario, etc.

Los *sujetos* se consideran también como *objetos del sistema* y un sujeto puede tener *derechos de acceder* a otro.

Los sujetos son entidades activas y los objetos son pasivos.

Una *capacidad* es una *señal:*

- La *posesión de una capacidad* por un sujeto le *confiere derechos de acceso* a un objeto.

Las capacidades no suelen ser modificadas pero suelen ser *reproducidas.*

Un *dominio de protección* define los *derechos de acceso* que un sujeto tiene a los distintos objetos del sistema:

- Es el *conjunto de capacidades* que pertenecen al *sujeto.*

Una *capacidad* es un *nombre protegido* para un objeto del sistema:

- El nombre es único en todo el sistema.
- Para tener acceso a un objeto determinado, un sujeto debe poseer una capacidad para hacerlo.

La *capacidad incluye* una instrucción de los *derechos de acceso* determinados que la capacidad le permite al sujeto respecto del objeto correspondiente.

La *creación de capacidades* es una función de rutinas de los S. O. cuidadosamente guardadas.

Lo normal es que *las capacidades no pueden ser modificadas* salvo para reducir los derechos de acceso establecidos.

Un sujeto con una capacidad puede *copiarla* o pasarla como un parámetro.

Luego de la creación de un objeto se crea una capacidad para ese objeto, que incluye todos los derechos de acceso al nuevo objeto.

El sujeto que crea la capacidad puede pasar copias de la capacidad a otros sujetos:

- La pueden usar o copiarla a otros sujetos:
  - Sin variantes.
  - Reduciendo (nunca incrementando) los derechos de acceso establecidos.

Si se han integrado *capacidades en el hardware de direccionamiento* del almacenamiento primario:

- Se las utiliza en cada referencia al mismo.
- Tenemos un *direccionamiento basado en la capacidad.*

En sistemas basados en capacidades se puede presentar el *problema del objeto perdido:*

- Si se destruye la última capacidad restante de un objeto, éste no podrá ser usado de ninguna manera.
- El sistema debe mantener siempre al menos una capacidad para cada objeto.

El control del *copiado y movimiento* de las capacidades es un *problema difícil*; generalmente el S. O. realiza la manipulación de capacidades en nombre de los usuarios.

La *revocación de las capacidades* pasadas a otro sujeto también puede complicarse:

- La capacidad pudo haber sido copiada muchas veces.
- Podría ser necesario revocar la capacidad:
  - De un sujeto determinado.
  - De cualquier otro sujeto que hubiera recibido de él esa capacidad.

Una técnica para la *revocación selectiva de las capacidades* es la siguiente:

- Todas las capacidades creadas a partir de una principal, apuntan al objeto a través de la capacidad principal.

14.11 Criptografía

El uso creciente de las *redes de computadoras* y la *importancia del trafico* cursado hace necesario *proteger a los datos [7, Deitel].*

La Oficina Nacional de Estándares de EE. UU. (NBS) ha adoptado la *norma de cifrado de datos* (DES) para la *transmisión de información* federal delicada.

La criptografía es el uso de la transformación de datos para hacerlos incomprensibles a todos, excepto a los usuarios a quienes están destinados.

El *problema de la intimidad* trata de cómo *evitar la obtención no autorizada* de información de un canal de comunicaciones.

El *problema de la autentificación* trata sobre *cómo evitar* que un oponente:

- Modifique una transmisión.
- Le introduzca datos falsos.

El *problema de la disputa* trata sobre *cómo proporcionar al receptor* de un mensaje pruebas legales de la *identidad del remitente*, que serían el equivalente electrónico de una firma escrita.

Un Sistema de Intimidad Criptográfica

El *remitente* desea transmitir cierto *mensaje no cifrado* (texto simple) a un *receptor* legítimo:

- La transmisión se producirá a través de un *canal inseguro:*
  - Se supone que podrá ser verificado o conectado mediante un espía.

El remitente pasa el *texto simple* a una *unidad de codificación* que lo transforma en un *texto cifrado o criptograma:*

- No es comprensible para el *espía.*
- Se transmite en forma *segura* por un *canal inseguro.*
- El receptor pasa el texto cifrado por una *unidad de descifrado* para regenerar el texto simple.

Criptoanálisis

Es el proceso de *intentar regenerar el texto simple a partir del texto cifrado*, pero *desconociendo la clave de ciframiento:*

- Es la tarea del espía o *criptoanalista:*
  - Si no lo logra, el sistema criptográfico es *seguro.*

Sistemas de Clave Pública

La *distribución de claves* de un sistema criptográfico debe hacerse por *canales muy seguros.*

Los *sistemas de clave pública* rodean el problema de distribución de claves:

- Las *funciones de cifrado y descifrado* están *separadas* y utilizan *distintas claves.*
- No es *computacionalmente* posible (en un tiempo *"razonable"*) determinar la *clave de desciframiento "D"* a partir de la *clave de ciframiento "C".*
- *"C"* puede hacerse *pública* sin comprometer la seguridad de *"D"*, que permanece *privada:*
  - Se simplifica el problema de la distribución de claves.

Firmas Digitales
Para que una *firma digital* sea aceptada como *sustituta de una firma escrita* debe ser:

- Fácil de autentificar (reconocer) por cualquiera.
- Producible únicamente por su autor.

En los *criptosistemas de clave pública* el procedimiento es:

- El *remitente* usa la *clave privada* para crear un *mensaje firmado.*
- El *receptor:*
  - Usa la *clave pública* del remitente para *descifrar el mensaje.*
  - Guarda el mensaje firmado para usarlo en caso de disputas.

Para *mayor seguridad* se podría actuar como sigue:

- El *remitente* puede *codificar el mensaje ya cifrado* utilizando la *clave pública del receptor.*
- La *clave privada del receptor* permite *recuperar el mensaje cifrado firmado.*
- La *clave pública del remitente* permite *recuperar el texto simple original.*

Aplicaciones
La *criptografía* es especialmente útil en los *sistemas multiusuario y en las redes de computadoras.*

Se debe utilizar para *proteger a las contraseñas*, almacenándolas cifradas.

Se puede utilizar también para *proteger todos los datos* almacenados en un sistema de computación; se debe considerar el tiempo de cifrado / descifrado.

También es aplicable en los *protocolos de redes de capas*, que ofrecen varios niveles de cifrado.

En el *cifrado de enlace* la red asume la responsabilidad de cifrado / descifrado de cada nodo:

- Los datos se transmiten cifrados entre los nodos.
- En cada nodo se descifran, se determina a dónde transmitirlos y se los vuelve a cifrar.

En el *cifrado punto a punto* un mensaje se cifra en su fuente y se descifra solo una vez, en su destino:

- Existen ciertas *limitaciones* tales como la *legibilidad de la dirección de destino* en cada nodo:
  - Debe ser legible para el encaminamiento del mensaje.
  - Ej.: sistemas de conmutación de paquetes de almacenamiento y reenvío con cifrado punto a punto; en este caso la *dirección de destino* asociada a un paquete *no puede ser cifrada.*

14.12 Penetración al Sistema Operativo

La *penetración definitiva* puede consistir en *cambiar el bit de estado de la máquina* del estado problema al estado supervisor; el intruso podrá así ejecutar *instrucciones privilegiadas* para obtener acceso a los recursos protegidos por el S. O. [7, Deitel].

Los *estudios de penetración* están diseñados para:

- Determinar si las defensas de un sistema contra ataques de usuarios no privilegiados son adecuadas.
- Descubrir deficiencias de diseño para corregirlas.

El *control de entrada / salida* es un área favorita para intentar la penetración a un sistema, ya que los canales de entrada / salida tienen acceso al almacenamiento primario y por consiguiente pueden modificar información importante.

Una de las *metas de las pruebas de penetración* consiste en estimar el *factor de trabajo de penetración:*

- Indicación de cuánto esfuerzo y recursos son necesarios para conseguir un acceso no autorizado a los recursos del sistema:
  - Debería ser tan grande que resulte *disuasivo.*

14.12.1 Principales Fallos Genéricos Funcionales de los Sistemas

Los principales fallos genéricos funcionales de los sistemas son los siguientes [7, Deitel]:

Autentificación:

- Los usuarios no pueden determinar si el hardware y el software con que funcionan son los que deben ser.
- Un intruso podría reemplazar un programa sin conocimiento del usuario.
- Un usuario puede inadvertidamente teclear una contraseña en un programa de entrada falso.

Cifrado:

- No se almacena cifrada la lista maestra de contraseñas.

Implementación:

- Implementación improcedente de un buen diseño de seguridad.

Confianza implícita:

- Una rutina supone que otra está funcionando correctamente cuando, de hecho, debería examinar los parámetros suministrados por la otra rutina.

Compartimiento implícito:

- El S. O. deposita inadvertidamente información importante del sistema en un espacio de direcciones del usuario.

Comunicación entre procesos:

- Usos inadecuados de los mecanismos de *send / receive* que pueden ser aprovechados por los intrusos.

Verificación de la legalidad:

- Validación insuficiente de los parámetros del usuario.

Desconexión de línea:

- Ante una desconexión de línea el S. O. debería:
  - Dar de baja al usuario (o los usuarios) de la línea.
  - Colocarlos en un estado tal que requieran la re - autorización para obtener nuevamente el control.

Descuido del operador:

- Un intruso podría engañar a un operador y hacer que le habilite determinados recursos.

Paso de parámetros por referencia en función de su valor:

- Es más seguro pasar los parámetros directamente en registros que tener los registros apuntando a las áreas que contienen los parámetros.
- El paso por referencia puede permitir que los parámetros, estando aún en el área del usuario, puedan ser modificados antes de ser usados por el sistema.

Contraseñas:

- No deben ser fácilmente deducibles u obtenibles mediante ensayos repetidos.

Entrampamiento al intruso:

- Los S. O. deben tener mecanismos de entrampamiento para atraer al intruso inexperto.

Privilegio:

- Cuando hay demasiados programas con demasiados privilegios se viola el *principio del menor privilegio.*

Confinamiento del programa:

- Un programa *"prestado"* de otro usuario puede actuar como un *"Caballo de Troya".*

Prohibiciones:

- Se advierte a los usuarios que no utilicen ciertas opciones porque los resultados podrían ser *"indeterminados"*, pero no se bloquea su uso, con lo que puede robar o alterar datos.

Residuos:

- Un intruso podría encontrar una lista de contraseñas con solo buscar en lugares tales como una *"papelera":*
  - Del sistema o física.
  - La información delicada debe ser sobrescrita o destruida antes de liberar o descartar el medio que ocupa.

Blindaje:

- Los intrusos pueden conectarse a una línea de transmisión sin hacer contacto físico:
  - Utilizan el campo inducido por la circulación de corriente en un cable.
  - Se previene con un adecuado blindaje eléctrico.

Valores de umbral:

- Si no se dispone de valores umbral, no habrá:
  - Límites al número de intentos fallidos de ingreso.
  - Bloqueos a nuevos intentos.
  - Comunicaciones al supervisor o administrador del sistema.

14.12.2 Ataques Genéricos a Sistemas Operativos

Los principales ataques genéricos a los S. O. son los siguientes [7, Deitel]:

Asincronismo:

- Se tienen procesos múltiples que progresan asincrónicamente.
- Un proceso podría modificar los parámetros ya validados por otro proceso pero aún no utilizados.
- Un proceso podría pasar valores malos a otro aún cuando el segundo realice una verificación extensa.

Rastreo:

- Un usuario revisa el sistema intentando localizar información privilegiada.

Entre líneas:

- Se utiliza una línea de comunicaciones mantenida por un usuario habilitado que está inactivo.

Código clandestino:

- Se modifica el S. O. bajo una presunta depuración pero se incorpora código que permite ingresos no autorizados.

Prohibición de acceso:

- Un usuario escribe un programa que bloquea el acceso o servicio a los usuarios legítimos mediante:
  - Caídas del sistema, ciclos infinitos, monopolio de recursos, etc.

Procesos sincronizados interactivos:

- Se utilizan las primitivas de sincronización del sistema para compartir y pasarse información entre sí.

Desconexión de línea:

- El intruso intenta acceder al trabajo de un usuario desconectado:
  - Luego de una desconexión de línea.
  - Antes de que el sistema reconozca la desconexión.

Disfraz:

- El intruso asume la identidad de un usuario legítimo luego de haber obtenido la identificación apropiada por medios clandestinos.

Ataque "nak":

- Si el S. O. permite a un usuario:
  - Interrumpir un proceso en ejecución mediante una *"tecla"* de *"reconocimiento negativo"*.
  - Realizar otra operación.
  - Reanudar el proceso interrumpido.
- Un intruso podría *"encontrar"* al sistema en un estado no protegido y hacerse con el control.

Engaño al operador:

- Con un engaño se hace realizar al operador una acción que comprometa la seguridad del sistema.

Parásito:

- Mediante equipamiento especial el intruso:
  - Intercepta los mensajes entre un usuario habilitado y el procesador.
  - Los modifica o reemplaza totalmente.

Caballo de Troya:

- El intruso coloca un código dentro del sistema que luego le permita accesos no autorizados.
- Puede permanecer en el sistema.
- Puede borrar todo rastro de sí mismo luego de la penetración.

Parámetros inesperados:

- El intruso suministra valores inesperados a una llamada al núcleo.
- Intenta aprovechar una debilidad de los mecanismos de verificación de la legalidad del S. O.

Parte III

Casos de Estudio

Capítulo 15

Algoritmos de Planificación del Procesador con P.O.O.

15.1 Introducción

Una de las muchas y muy variadas posibles aplicaciones de la *P.O.O. (programación orientada a objetos)* [2, Joyanes] está en el desarrollo de *algoritmos que implementen estrategias de administración de recursos* por parte del Sistema Operativo.

Como parte de las estrategias antes mencionadas, podemos considerar las de *administración o asignación del procesador*, es decir aquéllas según las cuales los S. O. seleccionan a *cuál de los procesos listos* para ser ejecutados en *ejecución concurrente*, le asignarán el procesador en un momento dado, es decir, a qué proceso darán la posibilidad de utilizar la CPU para ejecutar sus propias instrucciones; a esta decisión también se la conoce como *despacho del proceso* [23, Tanenbaum] y [7, Deitel].

15.2 Objetivo del Caso de Estudio

El objetivo del presente caso consistió en la realización de un *programa en Turbo C++* [12, Borland](**Procesos.cpp**) que *implementara las principales estrategias de asignación del procesador a procesos en ejecución concurrente.*

Asimismo también se estableció como objetivo del caso la inclusión en el programa de la posibilidad de grabar un *archivo de log histórico* (**Procesos.txt**), que incorporara para cada simulación efectuada, un resumen de las principales *conclusiones* referidas a las estrategias consideradas, lo que permite su análisis y discusión posterior.

15.3 Descripción del Problema Planteado

Una descripción detallada del problema planteado puede consultarse en el Capítulo *"Procesos y Administración del Procesador"*, siendo especialmente pertinentes los temas relacionados con *planificación del procesador.*

15.4 Descripción de los Algoritmos Utilizados

Al igual que en el apartado anterior, una descripción detallada de las *estrategias de planificación* y de los *algoritmos* para implementarlas se puede consultar en el Capítulo *"Procesos y Administración del Procesador"*.

15.5 Programa Desarrollado

El programa desarrollado (**Procesos.cpp**), codificado en *Turbo C++* utilizando *metodología de orientación a objetos ([2, Joyanes], [1, Joyanes], [3, Joyanes, Rodríquez y Fernández] y [15, Joyanes y Zahonero])*, implementa cuatro *estrategias de planificación del procesador*, a saber:

- **FIFO**: Primero en llegar, primero en ser despachado, es decir que los procesos son atendidos según su orden en la lista de procesos listos y una vez que reciben el procesador lo utilizan hasta que finalizan o hasta que se presenta una petición de entrada / salida requerida por el propio programa.

- **RR**: Round Robin: Los procesos son atendidos según su orden en la lista de procesos listos, pero disponen de un tiempo limitado (cuantum) del procesador, es decir que pueden ser interrumpidos por requerimientos propios de entrada / salida o por haber agotado su tiempo de procesador; obviamente que la otra causa de interrupción es la finalización del proceso.

- **HRN**: Los procesos son atendidos según su prioridad en la lista de procesos listos; la prioridad depende de la relación de respuesta: *(TE + TS) / TS*, donde *TE = Tiempo de Espera* y *TS = Tiempo de Servicio*; es decir que un proceso tiene mayor probabilidad de acceder a la cpu si ha hecho poco uso de ella; el tiempo de espera más el tiempo de servicio es el tiempo total que lleva el proceso *en el sistema*, es decir la cantidad de *ciclos de control* que se han contabilizado en la simulación, en tanto que el tiempo de servicio es el número de ciclos de control que el proceso ha utilizado; en este caso el proceso que dispone de la cpu puede ser interrumpido porque finaliza o porque requiere una operación de entrada / salida.

- **RNM**: Los procesos son atendidos según su nivel en la lista de procesos listos, la cual se divide en subcolas, cada una de ellas asociada a un nivel determinado, pero los procesos disponen de un tiempo limitado (cuantum) del procesador; si son interrumpidos por entrada / salida permanecen en la subcola del nivel donde están, pero si son interrumpidos por tiempo pasan a un nivel mayor, que tiene menor preferencia, que es atendido cuando ya no hay procesos listos en los niveles inferiores; de allí el nombre de *Retroalimentación de Niveles Múltiples*; de lo antes mencionado se desprende que un proceso puede ser interrumpido porque finaliza, porque agota su tiempo de procesador o porque requiere una operación de entrada / salida. En la implementación efectuada los niveles identificados con un número menor son los que tienen de hecho mayor prioridad y los identificados con un número mayor son los que reciben menor prioridad.

El *código del programa desarrollado* es el siguiente:

```
/* Simulación de estrategias de asignación del procesador */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <iostream.h>
#include <io.h>
FILE *textfile; /* Pointer al archivo usado */
// char regis[70]; /* Registro del archivo usado (para lectura) */
class Proc{
public:
/* identif: identificación del proceso */
char identif;
/* fin: estado de finalización del proceso */
/* 0: listo para ejecutar */
/* 1: interrumpido p/ entrada / salida */
/* 3: interrumpido p/ tiempo */
/* 2: terminado */
int fin;
/* cuenta: total de ciclos de control asignados al proceso */
int cuenta;
/* pri: prioridad del proceso */
int pri;
/* nivel: identificación de la sub-cola donde est el proceso */
int nivel;
Proc()
{ identif = 0; fin = 0; cuenta = 0; pri = 0; nivel = 0;}; // Inicialización de valores.
};
class procesos{
```

public:

int cc; // Número de ciclos de control.

int np; // Número de procesos.

char auxiden; // Variable auxiliar.

int auxfin; // Variable auxiliar.

int auxcuenta; // Variable auxiliar.

int auxpri; // Variable auxiliar.

int auxnivel; // Variable auxiliar.

int cambio; // Variable auxiliar.

Proc p[50]; // Vector (lista) de procesos.

void inicio(); // Acciones iniciales.

void introducir_datos(); // Carga de identificación de procesos.

void mostrar(); // Mostrar lista de procesos.

void fifo(); // Simulación según estrategia FIFO.

void rrobin(); // Simulación según estrategia RR (Round Robin).

void hrn(); // Simulación según estrategia HRN (Relación de Respuesta Máxima).

void rnm(); // Simulación según estrategia RNM (Retroalimentación de Niveles Múltiples).

void final(); // Acciones finales.

};

void procesos::inicio(){

cout << "**";

cout << "\n";

cout << "*Estrategias de asignación del procesador a procesos*";

cout << "\n";

cout << "*en ejecución concurrente. *";

cout << "\n";

cout << "**";

cout << "\n";

```
cout << "\n";
cout << "Introduzca el número de ciclos de control de la simulación.";
cout << "\n";
cout << "(Se sugiere entre 30 y 40 por cada 10 procesos): ";
cin >> cc;
cout << "\n";
cout << "Introduzca el número de procesos de la simulación.";
cout << "\n";
cout << "(El máximo permitido es 50): ";
cin >> np;
cout << "\n";
if (np > 50)
{np = 50;
cout << "El número de procesos se ha limitado a 50. \n";
cout << "\n";
}
/* Apertura del archivo usado para grabación */
cout << "Apertura del archivo resumen 'PROCESOS.TXT' para grabación. \n";
cout << "(Carpeta BIN de la carpeta TC). \n";
cout << "\n";
if ((textfile = fopen("procesos.txt", "a")) == NULL)
{printf("Error de apertura para grabación en el archivo 'procesos.txt'\n");
exit (0);
}
}
void procesos::introducir_datos(){
char opc;
cout << "\nSi desea introducir manualmente la identificación de los procesos\n";
```

```
cout << "tipee: 's', de lo contrario: 'n' y el sistema asumirá las identificaciones.\n";
cin >> opc;
if (opc == 's') {
for (register int i=0; i<np; i++)
{cout << "Introduzca la identificación del proceso p[" <<i << "]: ";
cin >> p[i].identif;
}
}
else {
p[0].identif = 'a';
p[1].identif = 'b';
p[2].identif = 'c';
p[3].identif = 'd';
p[4].identif = 'e';
p[5].identif = 'f';
p[6].identif = 'g';
p[7].identif = 'h';
p[8].identif = 'i';
p[9].identif = 'j';
p[10].identif = 'k';
p[11].identif = 'l';
p[12].identif = 'm';
p[13].identif = 'n';
p[14].identif = 'o';
p[15].identif = 'p';
p[16].identif = 'q';
p[17].identif = 'r';
p[18].identif = 's';
```

p[19].identif = 't';

p[20].identif = 'u';

p[21].identif = 'v';

p[22].identif = 'w';

p[23].identif = 'x';

p[24].identif = 'y';

p[25].identif = 'z';

p[26].identif = '1';

p[27].identif = '2';

p[28].identif = '3';

p[29].identif = '4';

p[30].identif = '5';

p[31].identif = '6';

p[32].identif = '7';

p[33].identif = '8';

p[34].identif = '9';

p[35].identif = 'A';

p[36].identif = 'B';

p[37].identif = 'C';

p[38].identif = 'D';

p[39].identif = 'E';

p[40].identif = 'F';

p[41].identif = 'G';

p[42].identif = 'H';

p[43].identif = 'I';

p[44].identif = 'J';

p[45].identif = 'K';

p[46].identif = 'L';

```
p[47].identif = ’M’;
p[48].identif = ’N’;
p[49].identif = ’O’;
}
for (register int i=0; i<np; i++)
{p[i].fin = 0;
p[i].cuenta = 0;
p[i].pri = 0;
p[i].nivel = 0;
}
}
void procesos::mostrar(){
printf(”\n”);
cout << ”Listado de procesos:” <<”\n”;
cout << ”*******************” <<”\n”;
cout << ”Estado: 0 -> Listo para ejecutar.” <<”\n”;
cout << ”Estado: 1 -> Interrumpido en espera de entrada / salida.” <<”\n”;
cout << ”Estado: 2 -> Proceso terminado.” <<”\n”;
cout << ”Estado: 3 -> Interrumpido por tiempo (no aplicable en FIFO ni en HRN).”
    <<”\n”;
for (register int i=0; i<np; i++)
{cout << ”Proceso p[” <<i << ”]: ” << p[i].identif<< ” Estado: ”<< p[i].fin<< ”
    Ciclos de control utilizados: ”<< p[i].cuenta<< ”\n”;
cout << ”Proceso p[” <<i << ”]: ” << p[i].identif<< ” Prioridad: ”<< p[i].pri<< ”
    Nivel: ”<< p[i].nivel;
cout <<”\n”;
}
}
void procesos::fifo(){
```

```
auxiden = 0; auxfin = 0; auxcuenta = 0; cambio = 0;
/* Grabación en el archivo usado */
fprintf(textfile, "%s\n", "*************************************************");
fprintf(textfile, "%s", "Simulación FIFO con ");
fprintf(textfile, "%i", cc);
fprintf(textfile, "%s\n", " ciclos de control.");
cout <<"\n";
cout <<"Secuencia de selección de procesos para su ejecución según FIFO:";
cout <<"\n";
cout <<"************************************************************";
cout <<"\n";
cout <<"Los procesos son atendidos según su orden en la lista de procesos listos:";
cout <<"\n" <<"\n";
for (register int j=0; j<cc; j++) // ***Realiza 'cc' ciclos de control***
{auxfin = 0;
for (register int m=0; m<np; m++)
{auxfin = auxfin + p[m].fin;}
if (auxfin == (np * 2))
{
cout <<"\n"<< "Todos los procesos han finalizado en "<< j << " ciclos de control.\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Todos los procesos han finalizado en ");
fprintf(textfile, "%i", j);
fprintf(textfile, "%s\n", " ciclos de control.");
j = cc;
auxfin = getch();
}
for (register int i=0; i<np; i++)
```

```
{if (p[i].fin == 0)
{
cout <<"Procesador asignado al proceso: "<< p[i].identif;
cout <<"\n";
p[i].cuenta = p[i].cuenta + 1;
p[i].fin = int(random(3)); // Determina próximo estado del proceso.
// cout <<"\n" <<p[i].identif <<p[i].fin <<"\n";
if (p[i].fin == 1)
cout <<"Proceso "' <<p[i].identif <<"' interrumpido por entrada / salida. "<< "\n";
else {if (p[i].fin == 2)
cout <<"Proceso "' <<p[i].identif <<"' finalizado. "<< "\n";
}
if (p[i].fin > 0)
{ // Intercambio de contexto.
cambio = cambio + 1;
auxiden = p[i].identif;
auxfin = p[i].fin;
auxcuenta = p[i].cuenta;
for (register int k=i; k<(np - 1); k++)
{p[k].identif = p[k+1].identif;
p[k].fin = p[k+1].fin;
p[k].cuenta = p[k+1].cuenta;
}
p[(np - 1)].identif = auxiden;
p[(np - 1)].fin = auxfin;
p[(np - 1)].cuenta = auxcuenta;
}
i = np;
```

```
auxfin = getch();
}
for (register int k=0; k<np; k++) // Determina si continúa la espera por entrada /
    salida.
{if (p[k].fin == 1) p[k].fin = int (random (2));
}
}
}
cout <<"\n"<< "***Se han producido "<< cambio<< " cambios de contexto.***"<<
    "\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Se han simulado ");
fprintf(textfile, "%i", np);
fprintf(textfile, "%s\n", " procesos concurrentes.");
fprintf(textfile, "%s", "Se han producido ");
fprintf(textfile, "%i", cambio);
fprintf(textfile, "%s\n", " cambios de contexto.");
auxfin = getch();
}
void procesos::rrobin(){
auxiden = 0; auxfin = 0; auxcuenta = 0; cambio = 0;
/* Grabación en el archivo usado */
fprintf(textfile, "%s\n", "************************************************");
fprintf(textfile, "%s", "Simulación RR - Round Robin con ");
fprintf(textfile, "%i", cc);
fprintf(textfile, "%s\n", " ciclos de control.");
cout <<"\n";
cout <<"Secuencia de selección de procesos para su ejecución según RR - Round Robin:";
cout <<"\n";
```

```
cout <<"********************************************************";
cout <<"\n";
cout <<"Los procesos son atendidos según su orden en la lista de procesos listos,\n";
cout <<"pero disponen de un tiempo limitado (cuantum) del procesador:";
cout <<"\n" <<"\n";
for (register int j=0; j<cc; j++) // ***Realiza 'cc' ciclos de control***
{auxfin = 0;
for (register int m=0; m<np; m++)
{if (p[m].fin != 2)
{auxfin = 1;
m = np;
}
}
if (auxfin == 0)
{
cout <<"\n"<< "Todos los procesos han finalizado en "<< j << " ciclos de control.\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Todos los procesos han finalizado en ");
fprintf(textfile, "%i", j);
fprintf(textfile, "%s\n", " ciclos de control.");
j = cc;
auxfin = getch();
}
for (register int i=0; i<np; i++)
{if (p[i].fin == 0)
{
cout <<"Procesador asignado al proceso: "<< p[i].identif;
cout <<"\n";
```

```
p[i].cuenta = p[i].cuenta + 1;
p[i].fin = int(random(4)); // Determina próximo estado del proceso.
// cout <<"\n" <<p[i].identif <<p[i].fin <<"\n";
if (p[i].fin == 0) p[i].fin = 3;
if (p[i].fin == 1)
cout <<"Proceso "' <<p[i].identif <<"' interrumpido por entrada / salida. "<< "\n";
else {if (p[i].fin == 2)
cout <<"Proceso "' <<p[i].identif <<"' finalizado. "<< "\n";
else {if (p[i].fin == 3)
cout <<"Proceso "' <<p[i].identif <<"' interrumpido por tiempo. "<< "\n";
}
}
if (p[i].fin >= 0)
{ // Intercambio de contexto.
cambio = cambio + 1;
auxiden = p[i].identif;
auxfin = p[i].fin;
auxcuenta = p[i].cuenta;
for (register int k=i; k<(np - 1); k++)
{p[k].identif = p[k+1].identif;
p[k].fin = p[k+1].fin;
p[k].cuenta = p[k+1].cuenta;
}
p[(np - 1)].identif = auxiden;
p[(np - 1)].fin = auxfin;
p[(np - 1)].cuenta = auxcuenta;
}
i = np;
```

```
auxfin = getch();
}
for (register int k=0; k<np; k++) // Determina si continúa la espera por entrada / salida.
{if (p[k].fin == 1) p[k].fin = int (random (2));
}
for (register int l=0; l<np; l++) // Determina si continúa la espera por tiempo.
{if (p[l].fin == 3)
{auxfin = int (random (4));
if (auxfin == 1) auxfin = 0;
else {if (auxfin == 2) auxfin = 3;}
p[l].fin = auxfin;
}
}
}
}
cout <<"\n"<< "***Se han producido "<< cambio<< " cambios de contexto.***"<< "\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Se han simulado ");
fprintf(textfile, "%i", np);
fprintf(textfile, "%s\n", " procesos concurrentes.");
fprintf(textfile, "%s", "Se han producido ");
fprintf(textfile, "%i", cambio);
fprintf(textfile, "%s\n", " cambios de contexto.");
auxfin = getch();
}
void procesos::hrn(){
auxiden = 0; auxfin = 0; auxcuenta = 0; auxpri = 0; cambio = 0;
```

```
/* Grabación en el archivo usado */
fprintf(textfile, "%s\n", "*****************************************************");
fprintf(textfile, "%s", "Simulación HRN con ");
fprintf(textfile, "%i", cc);
fprintf(textfile, "%s\n", " ciclos de control.");
cout <<"\n";
cout <<"Secuencia de selección de procesos para su ejecución según HRN:";
cout <<"\n";
cout <<"***************************************************************";
cout <<"\n";
cout <<"Los procesos son atendidos según su prioridad en la lista de procesos listos;\n";
cout <<"la prioridad depende de la relación de respuesta: (TE + TS) / TS, donde\n";
cout <<"TE = Tiempo de Espera y TS = Tiempo de Servicio:";
cout <<"\n" <<"\n";
for (register int j=0; j<cc; j++) // ***Realiza 'cc' ciclos de control***
{auxfin = 0;
for (register int m=0; m<np; m++)
{auxfin = auxfin + p[m].fin;}
if (auxfin == (np * 2))
{
cout <<"\n"<< "Todos los procesos han finalizado en "<< j << " ciclos de control.\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Todos los procesos han finalizado en ");
fprintf(textfile, "%i", j);
fprintf(textfile, "%s\n", " ciclos de control.");
j = cc;
auxfin = getch();
}
```

```
if (j == 0)
{for (register int z=0; z<np; z++)
{p[z].cuenta = 1;}
}
if (j < cc)
{
for (register int l=0; l<np; l++)
{p[l].pri = (j / p[l].cuenta);
}
}
if (auxpri == 1)
{
for (register int s=0; s<np; s++)
{for (register int t=s; t<(np - 1); t++)
{if (p[t+1].pri > p[t].pri)
{auxiden = p[t].identif;
auxfin = p[t].fin;
auxcuenta = p[t].cuenta;
auxpri = p[t].pri;
p[t].identif = p[t+1].identif;
p[t].fin = p[t+1].fin;
p[t].cuenta = p[t+1].cuenta;
p[t].pri = p[t+1].pri;
p[t+1].identif = auxiden;
p[t+1].fin = auxfin;
p[t+1].cuenta = auxcuenta;
p[t+1].pri = auxpri;
}
```

```
}
}
}
for (register int i=0; i<np; i++)
{if (p[i].fin == 0)
{auxpri = 0;
cout <<"Procesador asignado al proceso: "<< p[i].identif;
cout <<"\n";
p[i].cuenta = p[i].cuenta + 1;
p[i].fin = int(random(3)); // Determina próximo estado del proceso.
// cout <<"\n" <<p[i].identif <<p[i].fin <<"\n";
if (p[i].fin == 1)
cout <<"Proceso "' <<p[i].identif <<"' interrumpido por entrada / salida. "<< "\n";
else {if (p[i].fin == 2)
cout <<"Proceso "' <<p[i].identif <<"' finalizado. "<< "\n";
}
if (p[i].fin > 0)
{ // Intercambio de contexto.
cambio = cambio + 1;
auxiden = p[i].identif;
auxfin = p[i].fin;
auxcuenta = p[i].cuenta;
auxpri = p[i].pri;
for (register int k=i; k<(np - 1); k++)
{p[k].identif = p[k+1].identif;
p[k].fin = p[k+1].fin;
p[k].cuenta = p[k+1].cuenta;
p[k].pri = p[k+1].pri;
```

```
}
p[(np - 1)].identif = auxiden;
p[(np - 1)].fin = auxfin;
p[(np - 1)].cuenta = auxcuenta;
p[(np - 1)].pri = auxpri;
auxpri = 1; // Indica que hubo intercambio de contexto y debe reordenarse la lista de procesos según prioridades.
}
i = np;
auxfin = getch();
}
for (register int k=0; k<np; k++) // Determina si continúa la espera por entrada / salida.
{if (p[k].fin == 1) p[k].fin = int (random (2));
}
}
}
for (register int y=0; y<np; y++)
{p[y].cuenta = p[y].cuenta - 1;}
cout <<"\n"<< "***Se han producido "<< cambio<< " cambios de contexto.***"<< "\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Se han simulado ");
fprintf(textfile, "%i", np);
fprintf(textfile, "%s\n", " procesos concurrentes.");
fprintf(textfile, "%s", "Se han producido ");
fprintf(textfile, "%i", cambio);
fprintf(textfile, "%s\n", " cambios de contexto.");
auxfin = getch();
```

```
}
void procesos::rnm(){
auxiden = 0; auxfin = 0; auxcuenta = 0; auxpri = 0; auxnivel = 0; cambio = 0;
/* Grabación en el archivo usado */
fprintf(textfile, ”%s\n”, ”******************************************************”);
fprintf(textfile, ”%s”, ”Simulación RNM con ”);
fprintf(textfile, ”%i”, cc);
fprintf(textfile, ”%s\n”, ” ciclos de control.”);
cout <<”\n”;
cout <<”Secuencia de selección de procesos para su ejecución según RNM:”;
cout <<”\n”;
cout <<”******************************************************************”;
cout <<”\n”;
cout <<”Los procesos son atendidos según su nivel en la lista de procesos listos;\n”;
cout <<”pero disponen de un tiempo limitado (cuantum) del procesador;\n”;
cout <<”si son interrumpidos por entrada / salida permanecen en la subcola\n”;
cout <<”del nivel donde están, pero si son interrumpidos por tiempo pasan a\n”;
cout <<”un nivel superior, que es atendido cuando ya no hay procesos listos\n”;
cout <<”en los niveles inferiores; de allí el nombre de\n”;
cout <<”Retroalimentación de Niveles Múltiples:”;
cout <<”\n” <<”\n”;
for (register int j=0; j<cc; j++) // ***Realiza ’cc’ ciclos de control***
{auxfin = 0;
for (register int m=0; m<np; m++)
{if (p[m].fin != 2)
{auxfin = 1;
m = np;
}
```

```
}
if (auxfin == 0)
{
cout <<"\n"<< "Todos los procesos han finalizado en "<< j << " ciclos de control.\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Todos los procesos han finalizado en ");
fprintf(textfile, "%i", j);
fprintf(textfile, "%s\n", " ciclos de control.");
j = cc;
auxfin = getch();
}
if (auxpri == 1)
{
for (register int s=0; s<np; s++)
{for (register int t=s; t<(np - 1); t++)
{if (p[t+1].nivel < p[t].nivel)
{auxiden = p[t].identif;
auxfin = p[t].fin;
auxcuenta = p[t].cuenta;
auxpri = p[t].pri;
auxnivel = p[t].nivel;
p[t].identif = p[t+1].identif;
p[t].fin = p[t+1].fin;
p[t].cuenta = p[t+1].cuenta;
p[t].pri = p[t+1].pri;
p[t].nivel = p[t+1].nivel;
p[t+1].identif = auxiden;
p[t+1].fin = auxfin;
```

```
p[t+1].cuenta = auxcuenta;
p[t+1].pri = auxpri;
p[t+1].nivel = auxnivel;
}
}
}
}
for (register int i=0; i<np; i++)
{if (p[i].fin == 0)
{auxpri = 0;
cout <<"Procesador asignado al proceso: "<< p[i].identif;
cout <<"\n";
p[i].cuenta = p[i].cuenta + 1;
p[i].fin = int(random(4)); // Determina próximo estado del proceso.
// cout <<"\n" <<p[i].identif <<p[i].fin <<p[i].nivel <<"\n";
if (p[i].fin == 0) p[i].fin = 3;
if (p[i].fin == 1)
cout <<"Proceso "' <<p[i].identif <<"' interrumpido por entrada / salida. "<< "\n";
else {if (p[i].fin == 2)
cout <<"Proceso "' <<p[i].identif <<"' finalizado. "<< "\n";
else {if (p[i].fin == 3)
cout <<"Proceso "' <<p[i].identif <<"' interrumpido por tiempo. "<< "\n";
p[i].nivel = p[i].nivel + 1;
}
}
if (p[i].fin > 0)
{ // Intercambio de contexto.
cambio = cambio + 1;
```

```
auxiden = p[i].identif;
auxfin = p[i].fin;
auxcuenta = p[i].cuenta;
auxpri = p[i].pri;
auxnivel = p[i].nivel;
for (register int k=i; k<(np - 1); k++)
{p[k].identif = p[k+1].identif;
p[k].fin = p[k+1].fin;
p[k].cuenta = p[k+1].cuenta;
p[k].pri = p[k+1].pri;
p[k].nivel = p[k+1].nivel;
}
p[(np - 1)].identif = auxiden;
p[(np - 1)].fin = auxfin;
p[(np - 1)].cuenta = auxcuenta;
p[(np - 1)].pri = auxpri;
p[(np - 1)].nivel = auxnivel;
auxpri = 1; // Indica que hubo intercambio de contexto y debe reordenarse la lista de procesos según prioridades.
}
i = np;
auxfin = getch();
}
for (register int k=0; k<np; k++) // Determina si continúa la espera por entrada / salida.
{if (p[k].fin == 1) p[k].fin = int (random (2));
}
for (register int l=0; l<np; l++) // Determina si continúa la espera por tiempo.
{if (p[l].fin == 3)
```

```
{auxfin = int (random (4));
if (auxfin == 1) auxfin = 0;
else {if (auxfin == 2) auxfin = 3;}
p[l].fin = auxfin;
}
}
}
}
cout <<"\n"<< "***Se han producido "<< cambio<< " cambios de contexto.***"<<
    "\n";
/* Grabación en el archivo usado */
fprintf(textfile, "%s", "Se han simulado ");
fprintf(textfile, "%i", np);
fprintf(textfile, "%s\n", " procesos concurrentes.");
fprintf(textfile, "%s", "Se han producido ");
fprintf(textfile, "%i", cambio);
fprintf(textfile, "%s\n", " cambios de contexto.");
auxfin = getch();
}
void procesos::final(){
/* Cierre del archivo usado */
cout << "\nCierre del archivo resumen 'PROCESOS.TXT'. \n";
cout << "Se sugiere visualizar su contenido con el Notepad.exe de Windows. \n";
fclose(textfile);
}
void main(){
/* Variables */
procesos p1;
/* Código */
```

```
clrscr();
p1.inicio();
p1.introducir_datos();
p1.mostrar();
p1.fifo();
p1.mostrar();
p1.introducir_datos();
p1.mostrar();
p1.rrobin();
p1.mostrar();
p1.introducir_datos();
p1.mostrar();
p1.hrn();
p1.mostrar();
p1.introducir_datos();
p1.mostrar();
p1.rnm();
p1.mostrar();
p1.final();
getch();
}
```

15.6 Datos y Ejecuciones

Los datos esenciales para la simulación de las distintas estrategias se introducen por teclado, siendo opcional el ingreso de la identificación de los procesos, ya que la misma se puede generar automáticamente.

Los datos esenciales antes mencionados son el *número de ciclos de control* y el *número de procesos* que intervendrán en la *simulación.*

Los resultados detallados de las ejecuciones se muestran paso a paso en pantalla y un resumen de los mismos se graba en un archivo de texto (**Procesos.txt**), que puede ser visualizado con cualquier editor, por ejemplo el Notepad o el Wordpad de Windows.

El contenido del mencionado archivo luego de varias ejecuciones de evaluación puede consultarse en los Anexos; un detalle parcial es el siguiente:

****

Simulación FIFO con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 1 cambios de contexto.

****

Simulación RR - Round Robin con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación HRN con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 2 cambios de contexto.

****

Simulación RNM con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación FIFO con 100 ciclos de control.

Todos los procesos han finalizado en 14 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 10 cambios de contexto.

****

Simulación RR - Round Robin con 100 ciclos de control.

Todos los procesos han finalizado en 11 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 11 cambios de contexto.

****

Simulación HRN con 100 ciclos de control.

Todos los procesos han finalizado en 26 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 15 cambios de contexto.

****

Simulación RNM con 100 ciclos de control.

Todos los procesos han finalizado en 10 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 10 cambios de contexto.

****

15.7 Resultados y Conclusiones

La utilización de *P.O.O.* para la resolución del problema planteado ha resultado muy satisfactoria, destacándose las facilidades de la misma y del compilador utilizado *(Turbo C++)*.

Los *resultados obtenidos ratifican*, como era de esperarse, las *previsiones teóricas* en cuanto a las *diferencias* en atención a los procesos por parte de las distintas estrategias, ya sea en cuanto a la secuencia de *asignación del procesador* como en cuanto a la *cantidad de tiempo* (ciclos de control) asignados a los distintos procesos.

Las diferencias indicadas se hacen presentes *independientemente* de si se ha trabajado con un *número pequeño o grande*, tanto de *ciclos de control* como de *procesos.*

La modalidad implementada de mostrar los resultados paso a paso por pantalla permite observar el comportamiento de cada algoritmo y facilita la comprensión de su funcionamiento.

Como era de esperarse, se observa que los algoritmos que tienden a lograr una *mayor equitatividad en el uso del procesador* son los que generan una *mayor cantidad de cambios de contexto* (asignación del procesador a otro proceso distinto del que estaba siendo ejecutado y que se ha interrumpido).

Asimismo y a modo ilustrativo se han volcado en la Tabla 15.1 de la página 423 y en la Tabla 15.2 de la página 424 los datos del archivo resumen (**Procesos.txt**) como una planilla condensada de los mismos.

Estrategias	Ciclos de Control Solicitados	Finalizó Todos los Procesos	Ciclos de Control Utilizados	Procesos Concurrentes	Cambios de Contexto
FIFO	3	N	3	3	1
RR	3	N	3	3	3
HRN	3	N	3	3	2
RNM	3	N	3	3	3
FIFO	100	S	14	5	10
RR	100	S	11	5	11
HRN	100	S	26	5	15
RNM	100	S	10	5	10
FIFO	90	S	10	3	7
RR	90	S	10	3	9
HRN	90	S	22	3	13
RNM	90	S	14	3	14
FIFO	90	S	15	6	11
RR	90	S	32	6	32
HRN	90	S	20	6	13
RNM	90	S	12	6	12
FIFO	70	S	14	5	10
RR	70	S	11	5	11
HRN	70	S	26	5	15
RNM	70	S	10	5	10
FIFO	80	S	10	3	7
RR	80	S	10	3	9
HRN	80	S	22	3	13
RNM	80	S	14	3	14
FIFO	5	N	5	20	3
RR	5	N	5	20	5
HRN	5	N	5	20	3
RNM	5	N	5	20	5
FIFO	3	N	3	10	1
RR	3	N	3	10	3
HRN	3	N	3	10	2
RNM	3	N	3	10	3

Tabla 15.1: Estrategias de asignación del procesador en ejecución concurrente.

Estrategias	Ciclos de Control Solicitados	Finalizó Todos los Procesos	Ciclos de Control Utilizados	Procesos Concurrentes	Cambios de Contexto
FIFO	6	N	6	5	4
RR	6	N	6	5	6
HRN	6	N	6	5	5
RNM	6	N	6	5	6
FIFO	10	N	10	3	7
RR	10	N	10	3	9
HRN	10	N	10	3	6
RNM	10	N	10	3	10
FIFO	30	S	15	6	11
RR	30	N	30	6	30
HRN	30	S	15	6	13
RNM	30	S	16	6	16
FIFO	200	S	167	50	113
RR	200	S	174	50	174
HRN	200	S	161	50	98
RNM	200	S	198	50	198
FIFO	30	S	25	10	19
RR	30	N	30	10	30
HRN	30	N	30	10	20
RNM	30	N	30	10	30
FIFO	50	S	47	15	33
RR	50	N	50	15	50
HRN	50	N	50	15	31
RNM	50	S	46	15	46
FIFO	60	S	17	8	13
RR	60	S	32	8	32
HRN	60	S	29	8	19
RNM	60	S	28	8	28

Tabla 15.2: Estrategias de asignación del procesador en ejecución concurrente (continuación).

Capítulo 16

Paginación de Memoria Virtual con Sistemas Expertos

16.1 Introducción

Una de las muchas definiciones de **Sistema Experto** (SE) expresa que *es un sistema informático que simula el proceso de aprendizaje, memorización, razonamiento, comunicación y acción de un experto humano en una determinada rama de la ciencia, suministrando, de esta forma, un consultor que puede sustituirle con unas ciertas garantías de éxito.*

Otra definición [9, Castillo, Gutiérrez y Hadi] indica que un SE puede definirse como *un sistema informático (hardware y software) que simula a los expertos humanos en un área de especialización dada.*

Los SE constituyen un área dentro de un campo aún mayor llamado **Inteligencia Artificial** (IA), que según Barr y Feigenbaum *es la parte de la ciencia que se ocupa del diseño de sistemas de computación inteligentes, es decir, sistemas que exhiben las características que asociamos a la inteligencia en el comportamiento humano que se refiere a la comprensión del lenguaje, el aprendizaje, el razonamiento, la resolución de problemas, etc.*

La IA se propone combinar los métodos de búsqueda con grandes bases de conocimientos especializados, investigando cómo adquirir, representar, organizar y aplicar conocimiento a una gran variedad de problemas [13, Kvitca], es decir que la IA *es la ciencia que estudia cómo realizar programas que utilicen cada vez más tipos de conocimiento.*

La profundización en el estudio de estos conceptos lleva, entre otras muchas cuestiones, a la de considerar si los computadores pueden mostrar (o imitar) auténtica inteligencia, con todas las implicancias sociales y de otros tipos que ello comprende [16, Penrose].

Los principales *objetivos* de los SE pueden resumirse en los siguientes:

- Mejorar la calidad del conocimiento de los expertos humanos.

- Conseguir la supervivencia del conocimiento y que no desaparezca con la muerte física del experto humano.

- Multiplicar el número de expertos y por lo tanto, hacer más accesible el conocimiento existente.

Las principales *funciones* de los SE son las que a continuación se detallan:

- Adquirir conocimiento.
- Almacenar conocimiento.
- Razonar e inferir.
- Demandar nueva información.
- Aprender.
- Propagar incertidumbre.
- Asistir al experto a dar información coherente.

Las *razones* básicas para utilizar un SE son, entre otras, las que se mencionan a continuación:

- Posibilidad de utilizar personal no especializado para resolver problemas que requieren especialidad.
- Obtención de soluciones más rápidas.
- Obtención de soluciones más fiables.
- Reducción de costos.
- Eliminación de operaciones incómodas y monótonas.
- Escasez de expertos humanos.
- Acceso al conocimiento a poblaciones más amplias.
- Acceso a entornos donde la actuación humana directa resulta imposible.

Uno de los posibles campos de aplicación de los SE lo constituyen los Sistemas Operativos y el Software de Base y de Aplicación de los sistemas computacionales.

En dicho sentido, se deben considerar las funciones, especialmente de los Sistemas Operativos, como administradores de los recursos computacionales, tales como el procesador, la memoria principal, la memoria auxiliar, los periféricos, las operaciones de entrada / salida, etc., puesto que para el cumplimiento de dichas funciones emplean distintas estrategias que deben ser instrumentadas mediante módulos de software, siendo precisamente allí donde cabría considerar la posibilidad de utilizar SE para instrumentar las mismas y evaluar "inteligentemente" cuál sería la más conveniente en cada caso.

En este sentido es preciso señalar que no siempre los Sistemas Operativos actúan frente a la administración de recursos eligiendo la estrategia más adecuada para cada caso, sino que por el contrario utilizan siempre la misma estrategia, que obviamente no es en todos los casos la más adecuada.

16.2 Objetivo del Caso de Estudio

Los principales objetivos del caso de estudio desarrollado fueron los siguientes:

- Utilización de un software para SE, el **Expert System Builder (ESB)**, bajado de Internet, para generar un SE que permita obtener, dado un *"perfil (patrón) ideal de página de memoria virtual para el desalojo (pageout)"*, el grado cuantificado de ajuste de las páginas cargadas a ese "perfil", es decir el nivel de "acercamiento" a ese ideal planteado, con lo que se obtendría la página que más se ajusta a los criterios fijados para la operación de selección de la página que habrá de desalojarse de memoria principal (pageout).

- Desarrollo de un SE sencillo utilizando el software **Mathematica**, que evaluando una página de memoria virtual en particular, permita indicar bajo qué estrategias de reposición (desalojo) sería seleccionada dicha página para ser desalojada de la memoria principal.

Un objetivo mucho más ambicioso sería el de incorporar la utilización de SE en el núcleo de los Sistemas Operativos para que de manera más "inteligente" que la habitual, seleccionen en cada caso la mejor estrategia de paginación (desalojo o reposición de páginas), lo que conlleva a la elección de la página ideal para ser desalojada en cada ocasión en que esto sea requerido, optimizando así la utilización de los recursos computacionales, incluyendo no solo a la memoria principal sino también a las operaciones de entrada / salida que las operaciones de paginación implican y a los procesadores, que con esquemas "inteligentes" de paginación podrían dedicar mayor tiempo al código de los programas de aplicación y menor tiempo al código del Sistema Operativo, es decir a la gestión del cambio de contexto, la gestión de las operaciones de entrada / salida, etc.

16.3 Descripción del Problema Planteado

El problema planteado involucra la utilización de herramientas de SE para resolver problemas de *"administración de la memoria"* en los computadores, más concretamente, de *"administración del almacenamiento virtual"*, en lo referente a *"estrategias de reposición de páginas de la memoria real o principal".*

A los efectos de un adecuado encuadre en el tema, se resumen a continuación los principales conceptos relacionados con la *"Administración de la Memoria"* por parte de los Sistemas Operativos, pudiendo obtenerse más detalles en el capítulo del mismo nombre.

El estudio de la **administración de la memoria** comprende lo siguiente:

1. **Almacenamiento real:**
   (a) Organización y administración del almacenamiento.
   (b) Jerarquía de almacenamiento.
   (c) Estrategias de administración del almacenamiento.
   (d) Multiprogramación de partición fija.
   (e) Multiprogramación de partición variable.

   (f) Multiprogramación con intercambio de almacenamiento.

2. **Organización del almacenamiento virtual:**
   (a) Conceptos básicos de almacenamiento virtual.
   (b) Organización del almacenamiento de niveles múltiples.
   (c) Transformación de bloques.
   (d) Conceptos básicos de paginación.
   (e) Segmentación.
   (f) Sistemas de paginación / segmentación.

3. **Administración del almacenamiento virtual:**
   (a) Estrategias de administración del almacenamiento virtual.
   (b) Localidad.
   (c) Conjuntos de trabajo.
   (d) Paginación por demanda y paginación anticipada.
   (e) Liberación de página y tamaño de página.
   (f) Comportamiento de un programa en la paginación.

Un conjunto especialmente importante de aspectos de la **administración del almacenamiento virtual**[1] lo constituyen las **estrategias de administración del almacenamiento virtual.**

Las diferentes organizaciones de almacenamiento virtual generalmente implementadas son:

- Paginación.
- Segmentación.
- Combinación de segmentación y paginación.

Real	Sistemas de un solo usuario		
	Multiprogramación en memoria real	En partición fija	Absoluta
			Reubicable
		En partición variable	
Virtual	Multiprogramación en memoria virtual	Paginación pura	
		Segmentación pura	
		Combinado	

Tabla 16.1: Evolución en las organizaciones del almacenamiento.

Las estrategias para la administración de sistemas de almacenamiento virtual condicionan la conducta de los sistemas de almacenamiento virtual que operan según esas estrategias [23, Tanenbaum], las que se pueden clasificar de la siguiente manera:

[1]Ver Tabla 16.1 de la página 428 y Tabla 16.2 de la página 429.

Estrategias de búsqueda	Por demanda
	Anticipada
Estrategias de colocación	Mejor ajuste
	Primer ajuste
	Peor ajuste
Estrategias de reposición	Principio de optimización
	Al azar
	FIFO
	Pág. menos recientemente usada (LRU)
	Pág. menos frecuentemente usada (LFU)
	Pág. no usada recientemente (NUR)

Tabla 16.2: Estrategias de administración del almacenamiento virtual.

1. **"Estrategias de búsqueda":**
   - Tratan de los casos en que una página o segmento deben ser traídos del almacenamiento secundario al primario.
   - Las *"estrategias de búsqueda por demanda"* esperan a que se haga referencia a una página o segmento por un proceso antes de traerlos al almacenamiento primario.
   - Las *"estrategias de búsqueda anticipada"* intentan determinar por adelantado a qué páginas o segmentos hará referencia un proceso para traerlos al almacenamiento primario antes de ser explícitamente referenciados.
2. **"Estrategias de colocación":**
   - Tratan del lugar del almacenamiento primario donde se colocará una nueva página o segmento.
   - Los sistemas toman las decisiones de colocación de una forma trivial ya que una nueva página puede ser colocada dentro de cualquier marco de página disponible.
3. **"Estrategias de reposición":**
   - Tratan de la decisión de cuál página o segmento desplazar para hacer sitio a una nueva página o segmento cuando el almacenamiento primario está completamente comprometido.

En el conjunto de los distintos tipos de estrategias mencionadas se destacan las *"estrategias de reposición de página"* por su impacto directo en la performance (desempeño) de los procesos involucrados.

Las principales *"estrategias de reposición de página" [25, Tanenbaum]* son:

- El principio de optimización.

- Reposición de páginas al azar.
- Primero en entrar - primero en salir.
- Menos recientemente usada.
- Menos frecuentemente usada.
- No usada recientemente.

Las ideas básicas de las distintas *"estrategias de reposición"* son las que se detallan seguidamente:

El principio de optimización:

- El "principio de optimización" indica que para obtener un rendimiento óptimo, la página que se va a reponer es una que no se va a utilizar en el futuro durante el período de tiempo más largo.
- El problema es que no es factible predecir el futuro.

Reposición de página al azar:

- Consiste en escoger al azar la página que va a ser reemplazada.
- Todas las páginas del almacenamiento principal deben tener la misma probabilidad de ser reemplazadas.
- Debe poder seleccionar cualquier página, incluyendo la que va a ser referenciada a continuación (peor selección).
- Este esquema es raramente usado.

Reposición de página por el sistema de primero en entrar - primero en salir (FIFO):

- Se registra el momento en que cada página ingresa al almacenamiento primario.
- Para reemplazar una página, se selecciona aquella que ha estado más tiempo almacenada.
- Se presenta el inconveniente de que se pueden reemplazar páginas muy usadas, que serán llamadas de nuevo al almacenamiento primario casi de inmediato.

Un comentario especial merece la llamada *"anomalía FIFO"*:

- Belady, Nelson y Shedler descubrieron que con la reposición FIFO, ciertos patrones de referencias de páginas causan más fallos de páginas cuando se aumenta el número de marcos (celdas) de páginas asignados a un proceso: en esto consiste la *"anomalía FIFO"*.
- Esta anomalía contradice a la intuición[2].

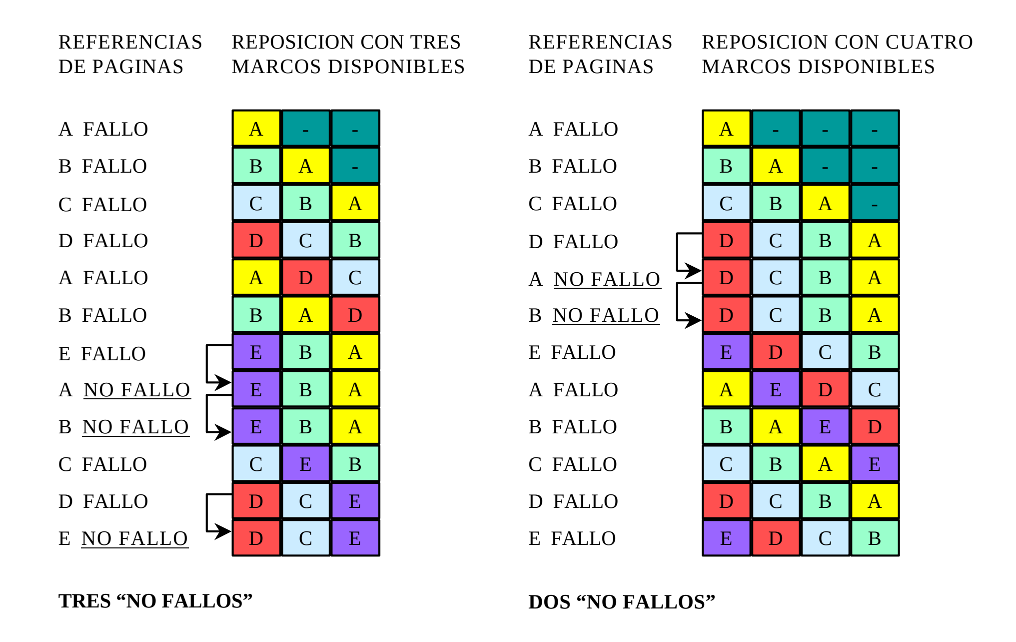


Figura 16.1: Ejemplo de anomalía FIFO.

Reposición de página menos - recientemente - usada (LRU):

- Esta estrategia selecciona para ser reemplazada la página que no ha sido usada durante el mayor período de tiempo.
- Se basa en la heurística de que el pasado reciente es un buen indicador del futuro próximo.
- Requiere que cada página reciba un *"sello de tiempo"* cada vez que se referencia:
  - Puede significar una sobrecarga adicional importante.
  - No se implementa frecuentemente.
- La página seleccionada para reemplazo podría ser la próxima en ser requerida, por lo cual habría que paginarla de nuevo al almacenamiento principal casi de inmediato.

Reposición de página menos - frecuentemente - usada (LFU):

- Acá interesa la intensidad de uso que haya tenido cada página.
- La página que será reemplazada es aquella que ha sido usada con menos frecuencia o que ha sido referida con menos intensidad.
- El inconveniente es que se puede seleccionar fácilmente para su reposición la página equivocada:

[2] Ver Figura 16.1 de la página 431 [7, Deitel].

- Ejemplo: La página de uso menos frecuente puede ser la página de entrada más reciente al almacenamiento principal, por lo que existe una alta probabilidad de que sea usada de inmediato.

Reposición de página no usada - recientemente (NUR):

- Presupone que las páginas que no han tenido uso reciente tienen poca probabilidad de ser usadas en el futuro próximo y pueden ser reemplazadas por otras nuevas.
- Es deseable reemplazar una página que no ha sido cambiada mientras estaba en el almacenamiento primario.
- La estrategia NUR se implementa con la adición de dos bits por página:
  - *"bit referenciado"*:
    - = 0 si la página no ha sido referenciada.
    - = 1 si la página ha sido referenciada.
  - *"bit modificado"* (también llamado *"bit sucio"*):
    - = 0 si la página no ha sido modificada.
    - = 1 si la página ha sido modificada.
- La selección de la página que será reemplazada comienza buscando una página que no ha sido referenciada, pero si no la encuentra, habrá que reemplazar una página que ha sido referenciada.
- Si una página no ha sido referenciada se comprueba si ha sido modificada o no:
  - Si no ha sido modificada se la reemplaza, ya que su reposición representa menos sobrecarga que la de una página modificada, puesto que en tal caso debería grabarse de nuevo en el almacenamiento secundario.
  - Si no se encuentra una página que no ha sido modificada, será reemplazada una página modificada.
- Con el transcurso del tiempo la mayoría de los *"bits referenciados"* serán activados y como consecuencia:
  - Se pierde la capacidad para distinguir las páginas más deseables para ser reemplazadas.
  - Para evitarlo se ajustan periódicamente todos los *"bits referenciados"* a "0":
    - Se logra un nuevo inicio.
    - Se vuelve vulnerable al reemplazo aún a las páginas activas, pero solo brevemente, mientras se reajustan los bits.
- Los *"bits modificados"* no se ajustan periódicamente según esta estrategia.

16.4 Descripción del Software Utilizado

Como primer parte del caso de estudio se ha utilizado un software para generación de SE, el **Expert System Builder (ESB)**, bajado de Internet, para generar un SE que permita obtener, dado un *"perfil (patrón) ideal de página de memoria virtual para el desalojo (pageout)"*, el grado cuantificado de ajuste de las páginas cargadas a ese "perfil", es decir el nivel de "acercamiento" a ese ideal planteado, con lo que se obtendría la página que más se ajusta a los criterios fijados para la operación de selección de la página que habrá de desalojarse de memoria principal (pageout).

El producto utilizado (**ESB**) **consta de tres partes** bien diferenciadas:

- **Question Editor**[3]:
  - El **Editor de Cuestiones** desarrolla las *"estructuras de las cuestiones"* para el SE y permite que esas cuestiones sean descriptas utilizando expresiones textuales o gráficas.
- **Knowledge Acquisition**[4]:
  - El **Programa de Adquisición del Conocimiento** permite al usuario ingresar datos acerca de cada registro que el sistema requiere *"conocer"*. Permite cargar la *"base de conocimientos"* para el SE.
- **User Interface**[5]:
  - La **Interfase de Usuario** intercala *"información del usuario"* y de la *"base de conocimientos"* y determina la solución más aproximada para el problema suministrado usando su *"motor de inferencia"*.

Asimismo el producto utilizado brinda un importante "help" en línea que resulta de mucha utilidad y la posibilidad de acceder a la Homepage correspondiente al producto mediante un vínculo con Internet.

Además se debe señalar que el producto considerado incluye varios ejemplos ilustrativos de las posibles utilizaciones del mismo, que resultan de interés conceptual y didáctico.

En otro orden de cosas se debe mencionar que el producto puede ser bajado de Internet sin cargo alguno.

Para la segunda parte del trabajo desarrollado se ha utilizado el software **Mathematica**, poderosa herramienta para cálculo simbólico, que en esta ocasión se ha utilizado teniendo presentes sus poderosas funciones de procesamiento de listas.

16.5 Descripción del Ejercicio Efectuado

Como ya se ha indicado anteriormente, en la primer parte del trabajo se ha utilizado el software **ESB**, para generar un SE que permita obtener, dado un *"perfil (patrón) ideal de*

[3] Ver Figura 16.2 de la página 434.
[4] Ver Figura 16.3 de la página 434.
[5] Ver Figura 16.4 de la página 435.

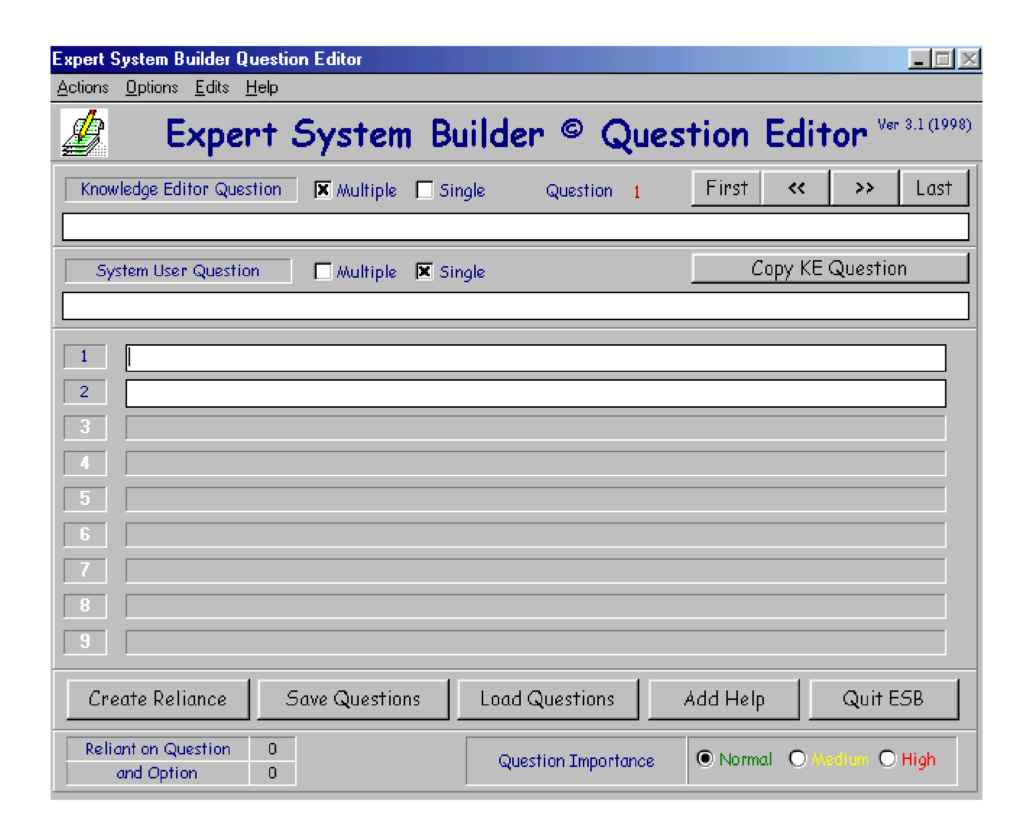


Figura 16.2: ESB Question Editor.

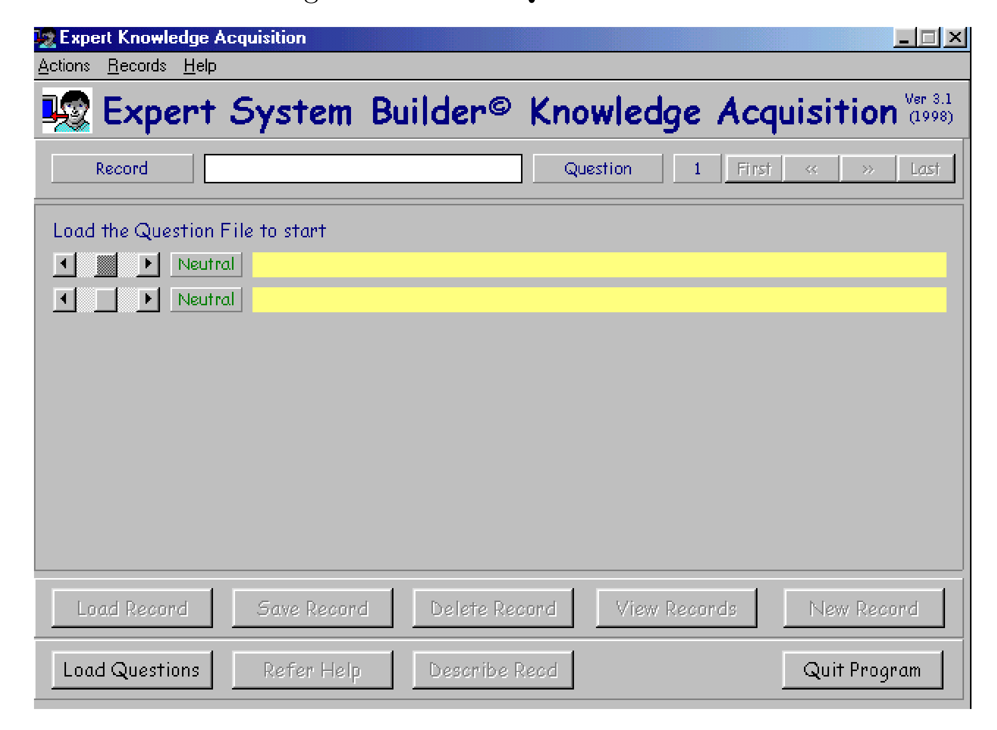


Figura 16.3: ESB Knowledge Acquisition.

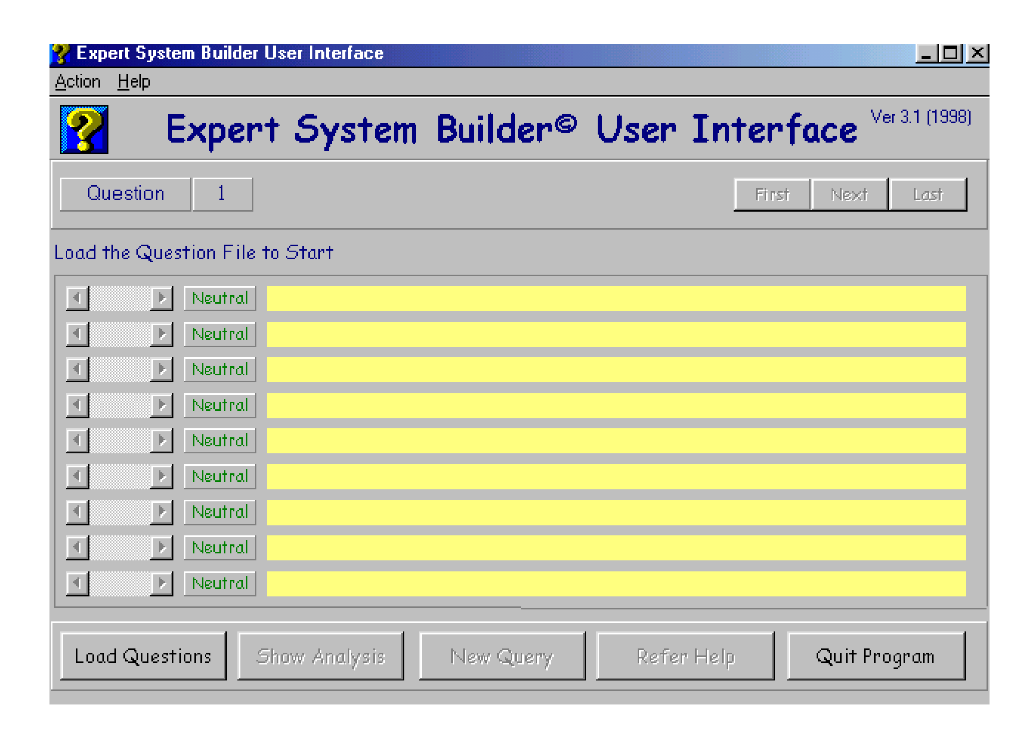


Figura 16.4: ESB User Interface.

página de memoria virtual para el desalojo (pageout)", el grado cuantificado de ajuste de las páginas cargadas a ese "perfil", es decir el nivel de "acercamiento" a ese ideal planteado, con lo que se obtendría la página que más se ajusta a los criterios fijados para la operación de selección de la página que habrá de desalojarse de memoria principal (pageout).

Primeramente y mediante la utilización del componente **Question Editor**, se han definido las *"cuestiones"* de interés, es decir las estructuras con las que habrá de trabajar el modelo; de esta forma se ha generado el archivo *"Pageout.qst"*, que luego es ingresado como entrada a los otros componentes del producto.

El contenido del archivo *"Pageout.qst"* utilizado es el siguiente:

Principio de optimización

Single

Principio de optimización - Tiempo estimado de no utilización mayor

Single

Menos de 5.000 miliseg.

Entre 5.000 y 10.000 miliseg.

Entre 10.001 y 20.000 miliseg.

Entre 20.001 y 40.000 miliseg.

Más de 40.000 miliseg.

Null

Null

Null

Null

4

0

0

NoTxt

NoPic

1

Reposición FIFO

Single

Reposición FIFO - Tiempo más prolongado en memoria principal

Single

Menos de 1.000 miliseg.

Entre 1.000 y 3.000 miliseg.

Entre 3.001 y 6.000 miliseg.

Entre 6.001 y 9.000 miliseg.

Entre 9.001 y 30.000 miliseg.

Más de 30.000 miliseg.

Null

Null

Null

5

0

0

NoTxt

NoPic

1

Reposición página menos recientemente usada (LRU)

Single

Reposición página menos recientemente usada (LRU) - Tiempo mayor de no uso

Single

Menos de 5.000 miliseg.

Entre 5.000 y 10.000 miliseg.

Entre 10.001 y 15.000 miliseg.

Entre 15.001 y 20.000 miliseg.

Entre 20.001 y 25.000 miliseg.

Entre 25.001 y 30.000 miliseg.

Entre 30.001 y 35.000 miliseg.

Entre 35.001 y 40.000 miliseg.

Más de 40.000 miliseg.

8

0

0

NoTxt

NoPic

2

Reposición página menos frecuentemente usada (LFU)

Single

Reposición página menos frecuentemente usada (LFU) - Menos frecuencia de uso

Single

Menos de 50 veces

Entre 50 y 100 veces

Entre 101 y 200 veces

Entre 201 y 300 veces

Más de 300 veces

Null

Null

Null

Null

4

0

0

NoTxt

NoPic

1.5

Reposición página no usada recientemente (NUR)

Single

Reposición página no usada recientemente (NUR) - Referencia y/o modificación

Single

No referenciada

Sí referenciada y no modificada

Sí referenciada y sí modificada

Null

Null

Null

Null

Null

Null

2

0

0

NoTxt

NoPic

2

Seguidamente y empleando el componente **Knowledge Acquisition** se ha cargado la *"base de conocimientos"*, que en este caso no es más que la definición de las características de las distintas páginas de memoria virtual residentes en un momento dado en memoria real (memoria principal) y que son factibles por lo tanto de ser sometidas a una operación de reposición o desalojo, es decir que pueden ser retiradas de la memoria principal para hacer lugar a otras páginas que se requiere alojar en la misma en un momento dado. Es preciso hacer notar que las características propias de cada página residente en memoria principal la hacen más o menos apropiada para ser seleccionada para el desalojo por las distintas estrategias de reposición, por lo cual la selección dependerá de la estrategia utilizada.

Para esta segunda etapa se ingresa el archivo generado anteriormente, es decir el archivo *"Pageout.qst"* y se genera el archivo *"Pageout.dat"*, el cual junto con el anterior es empleado por el *"motor de inferencia"* del modelo.

El contenido del archivo *"Pageout.dat"* utilizado es el siguiente:

Question 1=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=0 0 0 0 0 0 0 0 0

Question 2=0 0 0 0 0 0 0 0 0

Question 3=0 0 0 0 0 0 0 0 0

Question 4=0 0 0 0 0 0 0 0 0

Question 5=0 0 0 0 0 0 0 0 0

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 5=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 3=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=0 0 0 0 0 0 0 0 0

Question 3=0 0 0 0 0 0 0 0 0

Question 4=0 0 0 0 0 0 0 0 0

Question 5=0 0 0 0 0 0 0 0 0

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 4=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 -10 -10 -10 -10 -10 10 -10

Question 4=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 -10 -10 -10 -10 10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 -10 -10 -10 -10 10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 -10 -10 -10 -10 10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 4=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=10 -10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 -10 10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 1=-10 10 -10 -10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 5=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 1=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 2=-10 -10 -10 -10 10 -10 -10 -10 -10

Question 3=-10 -10 10 -10 -10 -10 -10 -10 -10

Question 4=-10 -10 -10 10 -10 -10 -10 -10 -10

Question 5=-10 10 -10 -10 -10 -10 -10 -10 -10

Por último y utilizando el componente **User Interface** se especifica el "perfil" de página que se toma como ideal para ser sometida a una operación de desalojo y el sistema produce una lista de las páginas ordenada de mayor a menor respecto de su grado de ajuste al perfil ingresado; esto permite ingresar distintos perfiles de páginas ideales para el desalojo, ya sea considerando las pautas de alguna de las estrategias de reposición o pautas mixtas, que no se ajusten estrechamente a ninguna estrategia y que tengan en cuenta más bien un conjunto de pautas, propias de las distintas estrategias, aunque sin ajustarse estrictamente a ninguna de ellas.

Lo señalado precedentemente tiene un gran valor didáctico y un potencial muy amplio de utilización en los Sistemas Operativos, para lo cual obviamente habrá que insertar el SE en el núcleo del Sistema Operativo, permitiendo que el mismo adecúe su estrategia de reposición de manera dinámica y flexible, atendiendo a situaciones externas ajenas a las estrategias de paginación.

Esta situación de ajuste dinámico de las estrategias de paginación en general y de reposición en particular puede resultar de mucho interés desde el punto de vista de optimizar los recursos computacionales logrando mayor performance (desempeño) permitiendo ajustar la estrategia de reposición a niveles superiores de planificación en el uso de recursos

por parte del Sistema Operativo, por ejemplo, cuando se pretende optimizar la planificación del procesador (o procesadores) mediante un esquema de planificación de niveles múltiples, es decir cuando para efectuar la asignación del procesador a los distintos procesos o cuando se debe seleccionar una página para reposición, no se tiene solo en cuenta una estrategia pura de asignación de procesador ni una estrategia pura de reposición de páginas, sino que por el contrario, ambas estrategias intercambian información y tratan de tomar decisiones teniendo en cuenta no solo la situación propia del recurso que deben administrar de manera directa (procesador o memoria) sino también la situación específica del otro recurso administrado por una estrategia de otro tipo, pero con el objetivo de lograr decisiones de administración de recursos que tengan una visión más amplia y un objetivo de optimización global y no solo local de cada recurso administrado.

16.6 Programas Desarrollados y Datos y Ejecuciones

Según se mencionó anteriormente, en la segunda parte del caso de estudio se ha desarrollado un SE sencillo utilizando el software **Mathematica**, que evaluando una página de memoria virtual en particular, permite indicar bajo qué estrategias de reposición (desalojo) sería seleccionada dicha página para ser desalojada de la memoria principal.

El programa desarrollado y algunas ejecuciones se encuentran en el archivo *"paginas.ma"*.

El contenido del archivo *"paginas.ma"* utilizado puede consultarse en los Anexos; un detalle parcial es el siguiente:

```
(* Ejemplo de Sistema Experto de paginación en memoria virtual *)

Paginacion[Prior_, Verosimilitudes_, DatosPagina_] :=

Module[{Prior1 = Prior, Posterior, i, j, aux, aux1,

n = Length[Prior], m = Length[DatosPagina]},

Print["******************************************"];

Print["* Análisis del encuadre de la página *"];

Print["* según las estrategias de paginación, *"];

Print["* lo que determinará la posibilidad de ser *"];

Print["* desalojada de la memoria principal *"];

Print["* según las distintas estrategias. *"];

Print["******************************************"];

Do[

Posterior = {};

Do[AppendTo[Posterior, If[DatosPagina[[j]] > 0,
```

```
(Verosimilitudes[i, Abs[DatosPagina[[j]]]]) * Prior1[[i]],
(1.0 - Verosimilitudes[[i, Abs[DatosPagina[[j]]]]])
* Prior1[[i]]]]
, {i, 1, n}];
aux = Sum[Posterior[[i]], {i, 1, n}];
Prior1 = Posterior / aux;
aux1 = Table[{Prior1[[i]], Estrategias[[i]]}, {i, 1, n}];
aux1 = Sort[aux1];
Print[" "]; Print[" "];
Print["Resultados tras la consideración de la pauta:"];
Print[" ", Pautas[[Abs[DatosPagina[[j]]]]]];
Print[" "];
Do[Print[aux1[[i, 2]], ": ", aux1[[i, 1]]]
, {i, n, 1, -1}];
, {j, 1, m}];
Print[" "]; Print[" "];
Print["**************************************************"];
Print["Resultado final:"];
Print[" La página tiene las siguientes probabilidades"];
Print[" de ser removida de la memoria principal según"];
Print[" las estrategias consideradas:"];
Print[" "];
Do[Print[aux1[[i, 2]], ": ", aux1[[i, 1]]]
, {i, n, 1, -1}];
Print["**************************************************"];
]
Estrategias = {"Tiempo estimado de no utilización mayor",
"Reposición FIFO - Tiempo mayor en memoria principal",
```

”Reposición LRU - Página menos recientemente usada”,

”Reposición LFU - Página menos frecuentemente usada”,

”Reposición NUR - Página no usada recientemente”};

Pautas = {”Tiempo hasta la próxima utilización: - 5.000 miliseg.”,

”Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.”,

”Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.”,

”Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.”,

”Tiempo hasta la próxima utilización: + 40.000 miliseg.”,

”Tiempo en mem. ppal.: - 1.000 miliseg.”,

”Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.”,

”Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.”,

”Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.”,

”Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.”,

”Tiempo en mem. ppal.: + 30.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.”,

”Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.”,

”Intensidad de utilización de la página: Menos de 50 veces”,

”Intensidad de utilización de la página: Entre 50 y 100 veces”,

”Intensidad de utilización de la página: Entre 101 y 200 veces”,

”Intensidad de utilización de la página: Entre 201 y 300 veces”,

”Intensidad de utilización de la página: Más de 300 veces”,

"Página no referenciada",

"Página sí referenciada y no modificada",

"Página sí referenciada y sí modificada"};

Prior = {0.1, 0.2, 0.25, 0.3, 0.5};

Verosimilitudes =

{{0.15, 0.25, 0.40, 0.60, 0.95,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.10, 0.20, 0.40, 0.60, 0.80, 1.00,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.85, 1.00,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

1.00, 0.70, 0.50, 0.25, 0.05,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

1.00, 0.50, 0.05}};

(* Ejemplo de página con mucho tiempo *)

(* estimado hasta la próxima utilización *)

DatosPagina =

{-1, -2, -3, -4, 5,

-6, -7, -8, -9, -10, -11,

-12, -13, -14, -15, -16, -17, -18, -19, -20,

-21, -22, -23, -24, -25,

-26, -27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.394967

Reposición LFU - Página menos frecuentemente usada: 0.23698

Reposición LRU - Página menos recientemente usada: 0.197484

Reposición FIFO - Tiempo mayor en memoria principal: 0.157987

Tiempo estimado de no utilización mayor: 0.0125818

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.547613

Reposición NUR - Página no usada recientemente: 0.180955

Reposición LFU - Página menos frecuentemente usada: 0.108573

Reposición LRU - Página menos recientemente usada: 0.0904773

Reposición FIFO - Tiempo mayor en memoria principal: 0.0723819

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Tiempo estimado de no utilización mayor: 0.551241

Reposición NUR - Página no usada recientemente: 0.182153

Reposición LFU - Página menos frecuentemente usada: 0.109292

Reposición LRU - Página menos recientemente usada: 0.0910766

Reposición FIFO - Tiempo mayor en memoria principal: 0.0662375

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Tiempo estimado de no utilización mayor: 0.558338

Reposición NUR - Página no usada recientemente: 0.184499

Reposición LFU - Página menos frecuentemente usada: 0.110699

Reposición LRU - Página menos recientemente usada: 0.0922493

Reposición FIFO - Tiempo mayor en memoria principal: 0.0542145

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Tiempo estimado de no utilización mayor: 0.570523

Reposición NUR - Página no usada recientemente: 0.188525

Reposición LFU - Página menos frecuentemente usada: 0.113115

Reposición LRU - Página menos recientemente usada: 0.0942625

Reposición FIFO - Tiempo mayor en memoria principal: 0.0335743

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Tiempo estimado de no utilización mayor: 0.582172

Reposición NUR - Página no usada recientemente: 0.192374

Reposición LFU - Página menos frecuentemente usada: 0.115425

Reposición LRU - Página menos recientemente usada: 0.0961871

Reposición FIFO - Tiempo mayor en memoria principal: 0.0138424

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Tiempo estimado de no utilización mayor: 0.588674

Reposición NUR - Página no usada recientemente: 0.194523

Reposición LFU - Página menos frecuentemente usada: 0.116714

Reposición LRU - Página menos recientemente usada: 0.0972615

Reposición FIFO - Tiempo mayor en memoria principal: 0.00282767

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Tiempo estimado de no utilización mayor: 0.590344

Reposición NUR - Página no usada recientemente: 0.195075

Reposición LFU - Página menos frecuentemente usada: 0.117045

Reposición LRU - Página menos recientemente usada: 0.0975373

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Tiempo estimado de no utilización mayor: 0.595625

Reposición NUR - Página no usada recientemente: 0.19682

Reposición LFU - Página menos frecuentemente usada: 0.118092

Reposición LRU - Página menos recientemente usada: 0.0894635

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Tiempo estimado de no utilización mayor: 0.60603

Reposición NUR - Página no usada recientemente: 0.200258

Reposición LFU - Página menos frecuentemente usada: 0.120155

Reposición LRU - Página menos recientemente usada: 0.0735567

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Tiempo estimado de no utilización mayor: 0.619376

Reposición NUR - Página no usada recientemente: 0.204668

Reposición LFU - Página menos frecuentemente usada: 0.122801

Reposición LRU - Página menos recientemente usada: 0.0531551

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Tiempo estimado de no utilización mayor: 0.632623

Reposición NUR - Página no usada recientemente: 0.209045

Reposición LFU - Página menos frecuentemente usada: 0.125427

Reposición LRU - Página menos recientemente usada: 0.0329042

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Tiempo estimado de no utilización mayor: 0.643096

Reposición NUR - Página no usada recientemente: 0.212506

Reposición LFU - Página menos frecuentemente usada: 0.127504

Reposición LRU - Página menos recientemente usada: 0.0168934

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Tiempo estimado de no utilización mayor: 0.649637

Reposición NUR - Página no usada recientemente: 0.214668

Reposición LFU - Página menos frecuentemente usada: 0.128801

Reposición LRU - Página menos recientemente usada: 0.00689504

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Tiempo estimado de no utilización mayor: 0.652774

Reposición NUR - Página no usada recientemente: 0.215704

Reposición LFU - Página menos frecuentemente usada: 0.129422

Reposición LRU - Página menos recientemente usada: 0.0020995

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.653939

Reposición NUR - Página no usada recientemente: 0.216089

Reposición LFU - Página menos frecuentemente usada: 0.129653

Reposición LRU - Página menos recientemente usada: 0.000318673

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.654147

Reposición NUR - Página no usada recientemente: 0.216158

Reposición LFU - Página menos frecuentemente usada: 0.129695

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Este programa ayuda a comprender las estrategias consideradas, ya que para una página con un "perfil" determinado, es decir con determinadas características, indica bajo qué estrategias es más probable que dicha página sea seleccionada para el desalojo.

En este caso se tiene en cuenta para el análisis la información propia de una página en particular y se despliega en pantalla cómo evolucionan los cálculos hasta concluir con todo el análisis de la información suministrada con cada página, teniendo en cuenta los valores de probabilidades asignados en el programa "a priori".

Además se han efectuado algunas modificaciones al programa inicial *"paginas.ma"* para permitirle realizar el análisis de un gran número de páginas generadas aleatoriamente (simulando de esta manera situaciones reales de páginas en un sistema de computación), para realizar con los resultados obtenidos un sencillo gráfico de número de veces que cada una de las estrategias consideradas resulta la más apropiada, según cada página considerada, para producir el desalojo de la misma. Este segundo programa y el resultado de algunas ejecuciones se encuentran en el archivo *"paginas2.ma"*, cuyo contenido se detalla seguidamente:

```
(* Ejemplo de Sistema Experto de paginación en memoria virtual *)

Paginacion[Prior_, Verosimilitudes_, DatosPagina_] :=

Module[{Prior1 = Prior, Posterior, i, j, aux, aux1,

n = Length[Prior], m = Length[DatosPagina]},

Do[

Posterior = {};

Do[AppendTo[Posterior, If[DatosPagina[[j]] > 0,

(Verosimilitudes[i, Abs[DatosPagina[[j]]]]) * Prior1[[i]],

(1.0 - Verosimilitudes[[i, Abs[DatosPagina[[j]]]]])

* Prior1[[i]]]]

, {i, 1, n}];

aux = Sum[Posterior[[i]], {i, 1, n}];

Prior1 = Posterior / aux;

aux1 = Table[{Prior1[[i]], Estrategias[[i]]}, {i, 1, n}];

aux1 = Sort[aux1];

, {j, 1, m}];

Do[
```

```
If[aux1[[n, 2]] == Estrategias[[k1]],
Contador[[k1]] = Contador[[k1]] + 1];
,{k1, 1, n}
];
]
Estrategias = {"Tiempo estimado de no utilización mayor",
"Reposición FIFO - Tiempo mayor en memoria principal",
"Reposición LRU - Página menos recientemente usada",
"Reposición LFU - Página menos frecuentemente usada",
"Reposición NUR - Página no usada recientemente"};
Contador = {0, 0, 0, 0, 0};
Pautas = {"Tiempo hasta la próxima utilización: - 5.000 miliseg.",
"Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.",
"Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.",
"Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.",
"Tiempo hasta la próxima utilización: + 40.000 miliseg.",
"Tiempo en mem. ppal.: - 1.000 miliseg.",
"Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.",
"Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.",
"Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.",
"Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.",
"Tiempo en mem. ppal.: + 30.000 miliseg.",
"Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.",
"Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.",
"Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.",
"Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.",
"Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.",
"Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.",
```

"Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.",

"Intensidad de utilización de la página: Menos de 50 veces",

"Intensidad de utilización de la página: Entre 50 y 100 veces",

"Intensidad de utilización de la página: Entre 101 y 200 veces",

"Intensidad de utilización de la página: Entre 201 y 300 veces",

"Intensidad de utilización de la página: Más de 300 veces",

"Página no referenciada",

"Página sí referenciada y no modificada",

"Página sí referenciada y sí modificada"};

Prior = {0.1, 0.2, 0.25, 0.3, 0.5};

Verosimilitudes =

{{0.15, 0.25, 0.40, 0.60, 0.95,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.10, 0.20, 0.40, 0.60, 0.80, 1.00,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.85, 1.00,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

```
{0.01, 0.01, 0.01, 0.01, 0.01,
0.01, 0.01, 0.01, 0.01, 0.01, 0.01,
0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,
1.00, 0.70, 0.50, 0.25, 0.05,
0.01, 0.01, 0.01},
{0.01, 0.01, 0.01, 0.01, 0.01,
0.01, 0.01, 0.01, 0.01, 0.01, 0.01,
0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,
0.01, 0.01, 0.01, 0.01, 0.01,
1.00, 0.50, 0.05}};
(* Generación de perfiles de páginas *)
Print["*******************************************"];
Print["* Análisis del encuadre de la página *"];
Print["* según las estrategias de paginación, *"];
Print["* lo que determinará la posibilidad de ser *"];
Print["* desalojada de la memoria principal *"];
Print["* según las distintas estrategias. *"];
Print["*******************************************"];
Print[" "];
Print[" Para las páginas consideradas, cuyo perfil "];
Print[" se genera aleatoriamente, la mayor probabilidad "];
Print[" de ser removidas de la memoria principal "];
Print[" corresponde a las estrategias:"];
Print[" "];
Contador = {0, 0, 0, 0, 0};
Do[
DatosPagina =
{-1, -2, -3, -4, -5,
```

```
-6, -7, -8, -9, -10, -11,
-12, -13, -14, -15, -16, -17, -18, -19, -20,
-21, -22, -23, -24, -25,
-26, -27, -28};
posinit=Random[Integer, {1, 5}];
DatosPagina[[posinit]] = DatosPagina[[posinit]] * (-1);
posinit=Random[Integer, {6, 11}];
DatosPagina[[posinit]] = DatosPagina[[posinit]] * (-1);
posi = posinit;
If[posi < 10, posinit = 12,
If[posi = 10, posinit=Random[Integer, {13, 17}],
If[posi = 11, posinit=Random[Integer, {18, 20}]
];
];
];
DatosPagina[[posinit]] = DatosPagina[[posinit]] * (-1);
posinit=Random[Integer, {21, 25}];
DatosPagina[[posinit]] = DatosPagina[[posinit]] * (-1);
posinit=Random[Integer, {26, 28}];
DatosPagina[[posinit]] = DatosPagina[[posinit]] * (-1);
Paginacion[Prior, Verosimilitudes, DatosPagina];
,{k, 1, 200} (* Se simulan 200 páginas *)
];
n = Length[Prior];
aux1 = Table[{Contador[[i]], Estrategias[[i]]}, {i, 1, n}];
aux1 = Sort[aux1];
Do[Print[aux1[[i, 2]], ": ", aux1[[i, 1]]]
, {i, n, 1, -1}
```

```
];
Print[" "];
Print["************************************************"];
Print[" "];
Print["Gráfico de Ocurrencia de Estrategias"];
Print["**********************************"];
Contador = Sort [Contador, Greater];
ListPlot[Contador,PlotStyle->
{RGBColor[1,0.6,0.3],PointSize[0.15]},
AxesLabel->{"Estrategias","Ocurrencias"},
AxesOrigin->{0,0}];
*****************************************
* Análisis del encuadre de la página *
* según las estrategias de paginación, *
* lo que determinará la posibilidad de ser *
* desalojada de la memoria principal *
* según las distintas estrategias. *
*****************************************
Para las páginas consideradas, cuyo perfil
se genera aleatoriamente, la mayor probabilidad
de ser removidas de la memoria principal
corresponde a las estrategias:
Tiempo estimado de no utilización mayor: 111
Reposición NUR - Página no usada recientemente: 66
Reposición LFU - Página menos frecuentemente usada: 23
Reposición LRU - Página menos recientemente usada: 0
Reposición FIFO - Tiempo mayor en memoria principal: 0
*************************************************
```

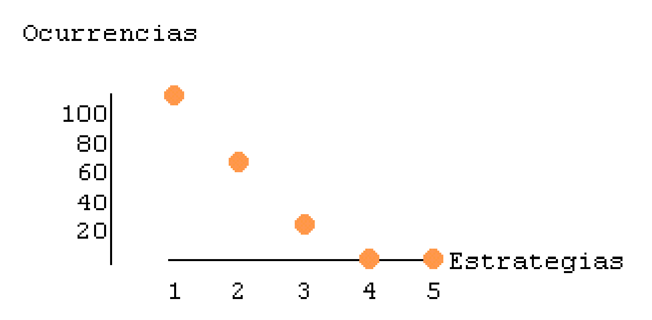


Figura 16.5: Número de ocurrencias según las estrategias de paginación.

Gráfico de Ocurrencia de Estrategias

****

Ver Figura 16.5 de la página 464.

16.7 Resultados y Conclusiones

Los resultados obtenidos al operar tanto con el SE generado con el ESB como con el SE generado con Mathematica, son muy ilustrativos y ayudan a la comprensión de los conceptos involucrados, tanto de SE como de estrategias de reposición en esquemas de memoria virtual.

Asimismo también se observa, como era de esperarse, que en todos los casos los resultados son coherentes con los conceptos teóricos y permiten reforzar el entendimiento de los mismos.

Además se hace notar que el archivo *"paginas2.ma"* indicado precedentemente contiene una simulación para doscientas páginas, pero también es han efectuado distintas pruebas con cantidades mayores, de varios miles de páginas, obteniéndose resultados de ocurrencias similares.

Los resultados obtenidos permiten también concluir lo que la teoría anticipa en el sentido de que no es eficiente desde este punto de vista un Sistema Operativo que siempre utilice la misma estrategia de reposición de páginas, puesto que para distintas configuraciones (características) de páginas las distintas estrategias producirían selecciones distintas, que obviamente repercutirían de manera diferente en la performance o desempeño global del sistema, puesto que las decisiones de reposición afectan también a la planificación de las operaciones de entrada / salida y también, aunque indirectamente, a las operaciones de planificación del procesador (o procesadores).

Por lo señalado anteriormente, sería conveniente que los Sistemas Operativos incorporaran en sus respectivos núcleos SE que efectúen las selecciones de páginas para reposición de manera "inteligente", con flexibilidad y adaptatividad a cada situación puntual y concreta, que además es naturalmente dinámica, especialmente en sistemas con multiprogra-

mación, multitarea y multiprocesamiento, donde se dan la concurrencia y el paralelismo en la ejecución de los distintos procesos.

Capítulo 17

Análisis del Rendimiento de un Subsistema de Disco de Una Petición a la Vez con Mathematica

17.1 Introducción

Para el desarrollo del presente caso de estudio se ha utilizado el software **Mathematica,** una muy poderosa herramienta para el *cálculo numérico* y el especialmente para el *cálculo simbólico* [5, Castillo, Iglesias, Gutiérrez, Alvarez y Cobo].

El software mencionado se ha utilizado para la realización de un *algoritmo que implemente el análisis de rendimiento de un subsistema de disco* ***de una petición a la vez,*** *utilizando Teoría de Colas.*

El estudio de las simulaciones efectuadas con el algoritmo señalado permite evaluar el comportamiento esperado de un subsistema de disco de una petición a la vez cuando es sometido a distintas *cargas de trabajo*, es decir a conjuntos de peticiones de operaciones de acceso a los discos que configuran *cantidades de trabajo* distintas a ser atendidas.

Los aspectos teóricos relacionados han sido desarrollados en los Capítulos *"Entrada / Salida"* y *"Modelado Analítico en Relación al Rendimiento"*.

17.2 Objetivo del Caso de Estudio

El objetivo del presente caso consistió en la realización de un *programa en Mathematica* (**Colas1pn.m**) que *implementara el algoritmo de análisis de rendimiento* para el caso señalado.

Asimismo también se estableció como objetivo del caso la inclusión en el programa de la posibilidad de generar *información detallada* respecto de los cálculos efectuados con las distintas simulaciones y un *análisis estadístico* de los resultados logrados, como así también un *gráfico* ilustrativo de los mismos, todo ello incluido en el archivo de ejecución y de resultados (**Colas1en.ma**).

17.3 Descripción del Problema Planteado

Una descripción detallada del problema planteado puede consultarse en el Capítulo *"Modelado Analítico en Relación al Rendimiento"*, siendo especialmente pertinentes los temas relacionados con *Proceso de Poisson* y *Análisis del Rendimiento de un Subsistema de Disco - Caso I.*

17.4 Descripción del Algoritmo Utilizado

Al igual que en el apartado anterior, una descripción detallada de los fundamentos teóricos y de las fórmulas matemáticas resultantes y aplicables a este caso, puede consultarse en el Capítulo *"Modelado Analítico en Relación al Rendimiento"*, siendo especialmente pertinente el tema sobre *Análisis del Rendimiento de un Subsistema de Disco - Caso I*, cuyo planteo resumido y esquema final de fórmulas aplicables se transcribe a continuación [7, Deitel]:

Caso I:

- El dispositivo de disco contiene un solo brazo.
- Solo puede dar servicio a una petición a la vez.
- La *tasa de servicio* es "μ".

Solución al caso I

"S_i" es el estado del sistema cuando hay *"i" peticiones* de disco al dispositivo de servicio de disco.

La *tasa de llegadas* de peticiones es independiente del estado del sistema:

- La *probabilidad de la transición* "S_i" $\rightarrow$ "S_{i+1}" en el siguiente intervalo de tiempo "Δt" es "$\lambda \Delta t$".

Se considera al sistema como un *proceso de nacimiento y muerte continuo de cadena sencilla y estados infinitos con:*

- $d_i = 0$ con i = 0.
- $d_i = \mu$ con i = 1, 2, 3, ...
- $b_i = \lambda$ con i = 0, 1, 2, ...
- Solo una petición puede ser servida en un momento dado y se sirve a una tasa μ.
- $\mu > \lambda$:
  - Asegura que la longitud de la cola de peticiones en espera *no crezca indefinidamente.*

Se utilizan las relaciones:

$P_{i+1} = (b_i / d_{i+1}) P_i$; i = 0, 1, 2, ...

$\sum_i$ P$_i$ = 1.

P$_1$ = (λ / μ) P$_0$.

P$_2$ = (λ / μ) P$_1$ = (λ / μ)^2 P$_0$.

P$_i$ = (λ / μ)^i P$_0$.

$\sum_i$ P$_i$ = 1 = $\sum_i$ (λ / μ)^i P$_0$ = 1/ [1 - (λ / μ)] P$_0$.

P$_0$ = 1 - (λ / μ): *probabilidad de que el sistema se encuentre ocioso.*

P$_i$ = (λ / μ)^i P$_0$ = [1 - (λ / μ)] (λ / μ)^i.   i = 0, 1, 2, ...

P$_i$ = [1 - (λ / μ)] (λ / μ)^i: *probabilidad que hayan "i" peticiones pendientes.*

El *número promedio de peticiones pendientes* es:

E(i) = $\sum_i$ i P$_i$ = $\sum_i$ i [1 - (λ / μ)] (λ / μ)^i = [1 - (λ / μ)] $\sum_i$ i (λ / μ)^i =

E(i) = [1 - (λ / μ)] (λ / μ) $\sum_i$ i (λ / μ)$^{i-1}$ =

E(i) = [1 - (λ / μ)] (λ / μ) {1 /[1 - (λ / μ)^2]} =

E(i) = (λ / μ) [1 - (λ / μ)$^{-1}$].

17.5 Programa Desarrollado

El programa desarrollado (**Colas1pn.m**), codificado en *Mathematica* utilizando especialmente las facilidades de cálculo numérico, análisis estadístico y graficación, implementa el ingreso interactivo de datos para la simulación, el cálculo de los resultados detallados y finales y el despliegue y almacenamiento de los mismos de una manera sencilla, guiada y explicativa.

El *código del programa desarrollado* es el siguiente:

```
(* Teoría de Colas *)

(* Análisis del rendimiento de un subsistema de disco *)

(* Caso 1: El subsistema de disco solo puede dar servicio a una petición a la vez. *)

(* Referencias y aclaraciones:

mu: Tasa promedio de servicio para un servidor.

mu = 1/Es(s)

Es(s): Tiempo de espera de servicio para un cliente.

lambda: Tasa promedio de llegadas de clientes al sistema de colas.

lambda = 1/Es(tau)
```

Es(tau): Tiempo de espera entre llegadas.

Es(tau) = 1/lambda

mu > lambda: Restricción para que la longitud de la cola de peticiones no crezca indefinidamente.

tau: Intervalo entre llegadas.

Pn: Probabilidad de que haya "n" clientes en el sistema de colas en estado estable.

P0: Probabilidad de que el sistema se encuentre ocioso.

P0 = 1-(lambda/mu)

Pri: Probabilidad de que hayan "i" peticiones pendientes.

Pri = (1-(lambda/mu))(lambda/mu)^i

(i=0,1,2,3,..)

Es(i): N° promedio de peticiones pendientes.

Es(i) = (lambda/mu)(1-(lambda/mu))^(-1) *)

BeginPackage["Colas1pn‘"]

<<Statistics‘LinearRegression‘

Colas1pn::usage=

"Colas1pn[mu,lambda,i]

Análisis del rendimiento de un subsistema

de disco que solo puede dar servicio a una

petición a la vez.

Colocar los valores máximos para mu, lambda

y el número de peticiones pendientes."

Colas1pn[inmu_,inlambda_,ini_]:=

Colas1pnAux[inmu,inlambda,ini];

Colas1pnAux[inmu_,inlambda_,ini_]:=

Module[{coef, Pri, Es, listacoef, listapetpend,

result},

Caso1[mu_,lambda_,i_]:=

If[(N[lambda/mu] >= 1.),

```
Print["Para que la cola no
crezca indefinidamente debe
ser Mu mayor que Lambda."],
{coef=N[lambda/mu,6],
Pri=N[(1.-lambda/mu)*(lambda/mu)^i,6],
Es=N[lambda/mu*(1.-lambda/mu)^(-1.),6],
Print["Análisis del rendimiento de un"]
Print["subsistema de disco."]
Print["Resultados del Caso 1:"]
Print["El subsistema de disco solo"]
Print["puede dar servicio a una"]
Print["petición a la vez."]
Print["Los valores de mu, lambda y el"]
Print["coeficiente son los siguientes:"]
Print[N[{mu,lambda,coef}]]
Print["La cola no crecerá"]
Print["indefinidamente debido a que"]
Print["Mu es mayor que Lambda."]
Print["Las probabilidades de tener"]
Print["0,1,2,3,4,...peticiones"]
Print["pendientes son las siguientes:"]
Print[N[Pri]]
Print["El promedio de peticiones"]
Print["pendientes es el siguiente:"]
Print[N[Es]]
Print["****************************"]
AppendTo[listacoef,coef]
AppendTo[listapetpend,Es]
```

```
AppendTo[result,{coef,Es}]
}
];
Modelo1[mum_,lambdam_,im_]:=
If[(N[lambdam/mum] >= 1.),
Throw["Para que la cola no
crezca indefinidamente debe
ser Mu mayor que Lambda."],
For[i=10, i<=mum, i+=10,
For[j=1, j<=lambdam, j+=2,
Caso1[i,j,{Range[0,im,1]}]
];
];
];
listacoef={};
listapetpend={};
result={};
(* Colocar los valores máximos para mu, lambda *)
(* y el número de peticiones pendientes. *)
Modelo1[inmu,inlambda,ini];
Print[" "];
Print["****************************"];
Print["******* Resúmen final *******"];
Print["****************************"];
Print["Lista de coeficientes lambda/mu:"];
Print[listacoef];
Print["Promedio de peticiones"];
Print["pendientes para cada coeficiente:"];
```

```
Print[listapetpend];
Print["Pares (coeficientes,promedio):"];
Print[result];
Print["Gráfico de promedio de peticiones"];
Print["pendientes para cada coeficiente y"];
Print["Análisis de regresión lineal:"];
ListPlot[result,PlotStyle->
{RGBColor[1,0.6,0.3],PointSize[0.023]},
AxesLabel->{Coeficientes,Esperas},
AxesOrigin->{0,0}];
Regress[result,{1,x,x^2,x^3},x,
OutputList->{BestFit,
BestFitCoefficients,ANOVATable,
ConfidenceIntervalTable,
CovarianceMatrix,CorrelationMatrix,
EstimatedVariance,FitResiduals,
ParameterTable,PredictedResponse,
RSquared,AdjustedRSquared}]
]
EndPackage[];
```

17.6 Datos y Ejecuciones

Los datos para la simulación se introducen por teclado, siendo ellos los valores de μ, λ y el número de peticiones pendientes.

Los resultados detallados de las ejecuciones se muestran paso a paso en pantalla y pueden ser grabados en el mismo archivo de datos y ejecuciones, que en este caso ha sido el archivo **Colas1en.ma.**

El contenido del mencionado archivo luego de varias ejecuciones de evaluación puede consultarse en los Anexos; un detalle parcial es el siguiente, aclarándose que no se ha incluido en el contenido mostrado el análisis estadístico detallado:

```
<<Examples`Colas1pn`
```

? Colas1pn

Colas1pn[mu,lambda,i]

Análisis del rendimiento de un subsistema de disco que solo puede dar servicio a una petición a la vez.

Colocar los valores máximos para mu, lambda y el número de peticiones pendientes.

Colas1pn[20,10,3]

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009}}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7, 0.21, 0.063, 0.0189}}

El promedio de peticiones pendientes es el siguiente:

0.428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.5, 0.25, 0.125, 0.0625}}

El promedio de peticiones pendientes es el siguiente:

1.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.3, 0.21, 0.147, 0.1029}}

El promedio de peticiones pendientes es el siguiente:

2.33333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.1, 0.09, 0.081, 0.0729}}

El promedio de peticiones pendientes es el siguiente:

9.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.95, 0.0475, 0.002375, 0.00011875}}

El promedio de peticiones pendientes es el siguiente:

0.0526316

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.85, 0.1275, 0.019125, 0.00286875}}

El promedio de peticiones pendientes es el siguiente:

0.176471

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.75, 0.1875, 0.046875, 0.0117188}}

El promedio de peticiones pendientes es el siguiente:

0.333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.65, 0.2275, 0.079625, 0.0278688}}

El promedio de peticiones pendientes es el siguiente:

0.538462

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.55, 0.2475, 0.111375, 0.0501187}}

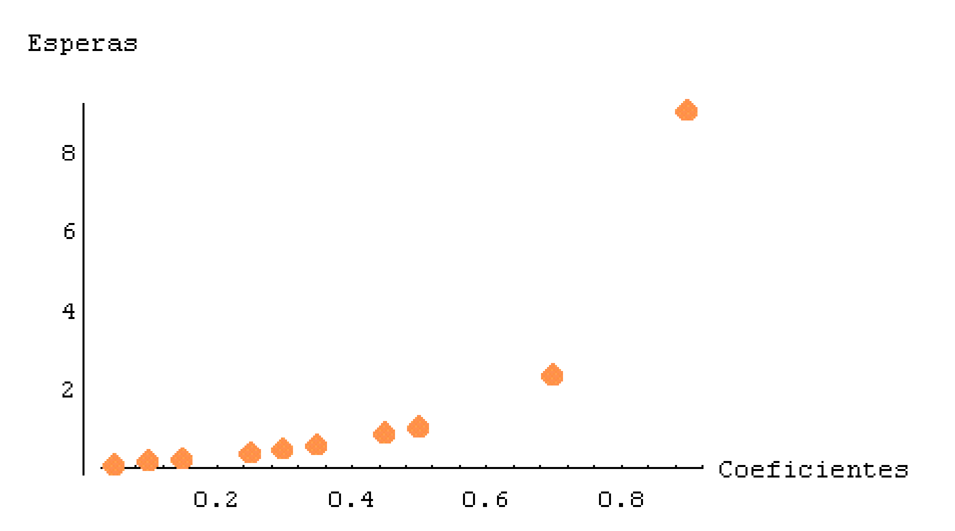


Figura 17.1: Promedio de peticiones pendientes.

El promedio de peticiones pendientes es el siguiente:

0.818182

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45}

Promedio de peticiones pendientes para cada coeficiente:

{0.111111, 0.428571, 1., 2.33333, 9., 0.0526316, 0.176471, 0.333333, 0.538462, 0.818182}

Pares (coeficientes,promedio):

{{0.1, 0.111111}, {0.3, 0.428571}, {0.5, 1.}, {0.7, 2.33333}, {0.9, 9.}, {0.05, 0.0526316}, {0.15, 0.176471}, {0.25, 0.333333}, {0.35, 0.538462}, {0.45, 0.818182}}

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

{Statistics‘LinearRegression‘BestFit ->

-0.706132 + 12.4786 x - 41.07 x^2 + 43.3673 x^3 ,

Statistics‘LinearRegression‘BestFitCoefficients ->

{-0.706132, 12.4786, -41.07, 43.3673}}

17.7 Resultados y Conclusiones

La utilización de *Mathematica* para la resolución del problema planteado ha resultado muy satisfactoria, destacándose las facilidades y potencia del producto.

Los *resultados obtenidos ratifican*, como era de esperarse, las *previsiones teóricas* en cuanto a las *diferencias* en tiempos en cola de espera en disco de las distintas peticiones, según las distintas *cargas de trabajo* simuladas.

La modalidad implementada de mostrar los resultados paso a paso por pantalla permite observar el comportamiento del algoritmo y facilita la comprensión de su funcionamiento.

Como era de esperarse, se observa que en todos los casos, *la forma de la curva de los gráficos* obtenidos es similar, apreciándose un muy buen ajuste de los valores obtenidos (y graficados) a un polinomio de tercer orden.

Asimismo se observa un *fuerte impacto en los tiempos de espera* a partir de cierto nivel de *carga de trabajo*, expresada según los coeficientes considerados.

Capítulo 18

Análisis del Rendimiento de un Subsistema de Disco de Varias Peticiones a la Vez con Mathematica

18.1 Introducción

Para el desarrollo del presente caso de estudio se ha utilizado el software **Mathematica,** una muy poderosa herramienta para el *cálculo numérico* y el especialmente para el *cálculo simbólico* [5, Castillo, Iglesias, Gutiérrez, Alvarez y Cobo].

El software mencionado se ha utilizado para la realización de un *algoritmo que implemente el análisis de rendimiento de un subsistema de disco* ***de varias peticiones a la vez,*** *utilizando Teoría de Colas.*

El estudio de las simulaciones efectuadas con el algoritmo señalado permite evaluar el comportamiento esperado de un subsistema de disco de varias peticiones a la vez cuando es sometido a distintas *cargas de trabajo*, es decir a conjuntos de peticiones de operaciones de acceso a los discos que configuran *cantidades de trabajo* distintas a ser atendidas.

La posibilidad de atender varias peticiones en paralelo requiere que el subsistema de disco disponga de un número elevado de brazos de acceso, que además sean independientes en sus movimientos, de tal manera que cada uno de ellos pueda atender a un petición.

Los aspectos teóricos relacionados han sido desarrollados en los Capítulos *"Entrada / Salida"* y *"Modelado Analítico en Relación al Rendimiento"*.

18.2 Objetivo del Caso de Estudio

El objetivo del presente caso consistió en la realización de un *programa en Mathematica* (**Colas2pn.m**) que *implementara el algoritmo de análisis de rendimiento* para el caso señalado.

Asimismo también se estableció como objetivo del caso la inclusión en el programa de la posibilidad de generar *información detallada* respecto de los cálculos efectuados con las distintas simulaciones y un *análisis estadístico* de los resultados logrados, como así

también un *gráfico* ilustrativo de los mismos, todo ello incluido en el archivo de ejecución y de resultados (**Colas2en.ma**).

18.3 Descripción del Problema Planteado

Una descripción detallada del problema planteado puede consultarse en el Capítulo *"Modelado Analítico en Relación al Rendimiento"*, siendo especialmente pertinentes los temas relacionados con *Proceso de Poisson* y *Análisis del Rendimiento de un Subsistema de Disco - Caso II.*

18.4 Descripción del Algoritmo Utilizado

Al igual que en el apartado anterior, una descripción detallada de los fundamentos teóricos y de las fórmulas matemáticas resultantes y aplicables a este caso, puede consultarse en el Capítulo *"Modelado Analítico en Relación al Rendimiento"*, siendo especialmente pertinente el tema sobre *Análisis del Rendimiento de un Subsistema de Disco - Caso II*, cuyo planteo resumido y esquema final de fórmulas aplicables se transcribe a continuación [7, Deitel]:

Caso II:

- El dispositivo de disco contiene gran número de brazos móviles.
- Cada brazo puede dar servicio a una petición de disco a la misma tasa "μ".
- Se supone que un número infinito de peticiones pueden recibir servicio en paralelo.

Solución al caso II

Con *"i" peticiones* siendo servidas:

- La *probabilidad de que una petición en particular acabe siendo servida* dentro del siguiente *"Δt"* es *"$\mu\Delta t$".*
- La *probabilidad de que exactamente una petición cualquiera acabe* es *"$i\mu\Delta t$"* (buena aproximación de primer orden).
- Cualquiera de las *"i" peticiones* puede terminar y provocar un *cambio de estado.*

El sistema se ve como un *proceso de nacimiento y muerte continuo de cadena sencilla y de estados infinitos con:*

- $b_i = \lambda$ con i = 0, 1, 2, ...
- $d_i = 0$ con i = 0.
- $d_i = i\mu$ con i = 1, 2, 3, ...

Ningún cliente tiene que esperar ya que *se suponen infinitos servidores en paralelo.*

Se utilizan las relaciones:

$P_{i+1} = (b_i / d_{i+1}) P_i; \qquad i = 0, 1, 2, ...$

$\sum_i P_i = 1.$

$P_1 = (\lambda / \mu) P_0.$

$P_2 = (\lambda / 2\mu) P_1 = (1 / 2) (\lambda / \mu)^2 P_0.$

$P_3 = (\lambda / 3\mu) P_2 = (1 / (3 . 2)) (\lambda / \mu)^3 P_0.$

$P_i = (1 / i!) (\lambda / \mu)^i P_0.$

$\sum_i P_i = 1 = \sum_i (1 / i!) (\lambda / \mu)^i P_0.$

$\sum_n (x^n / n!) = e^x.$

$\sum_i P_i = 1 = \sum_i (1 / i!) (\lambda / \mu)^i P_0 = e^{\lambda/\mu} P_0.$

$\mathbf{P}_0 = e^{-\lambda/\mu}$.

$\mathbf{P}_i = (\lambda / \mu)^i [(e^{-\lambda/\mu}) / i!].$

$E(i) = \sum_i iP_i = \sum_i i (\lambda / \mu)^i [(e^{-\lambda/\mu}) / i!] = (e^{-\lambda/\mu}) \sum_i i (\lambda / \mu)^i (1 / i!) =$

$E(i) = (e^{-\lambda/\mu}) \sum_i (\lambda / \mu) (\lambda / \mu)^{i-1} [1 / (i - 1)!] =$

$E(i) = (e^{-\lambda/\mu}) (\lambda / \mu) \sum_i [1 / (i - 1)!] (\lambda / \mu)^{i-1} =$

$E(i) = (e^{-\lambda/\mu}) (\lambda / \mu) (e^{\lambda/\mu}) =$

$\mathbf{E(i)} = (\lambda / \mu).$

Conclusiones:

- En el sistema de *un solo servidor*, si una petición que llega encuentra ocupado el dispositivo de disco, debe esperar.

- En el sistema de *servidores infinitos*, las peticiones que llegan siempre entran al servicio de inmediato.

- En el sistema de *un solo servidor:*

  - A medida que λ tiende a μ la probabilidad de que el sistema se encuentre ocioso decrece rápidamente:
    - Las peticiones que llegan esperan.
  - El número promedio de peticiones pendientes crece con rapidez.

- En el sistema de *servidores infinitos:*

  - El número promedio de peticiones pendientes tiende a 1.

18.5 Programa Desarrollado

El programa desarrollado (**Colas2pn.m**), codificado en *Mathematica* utilizando especialmente las facilidades de cálculo numérico, análisis estadístico y graficación, implementa el ingreso interactivo de datos para la simulación, el cálculo de los resultados detallados y finales y el despliegue y almacenamiento de los mismos de una manera sencilla, guiada y explicativa.

El *código del programa desarrollado* es el siguiente:

```
(* TEORIA DE COLAS *)

(* Análisis del rendimiento de un subsistema de disco *)

(* Caso 2: El subsistema de disco, que posee varios brazos móviles, puede dar

servicio a una petición mediante cada uno de ellos a la vez. *)

(* Referencias y aclaraciones:

mu: Tasa promedio de servicio para c/u de los brazos móviles.

mu = 1/Es(s)

Es(s): Tiempo de espera de servicio para un cliente.

lambda: Tasa promedio de llegadas de clientes al sistema de colas.

lambda = 1/Es(tau)

Es(tau): Tiempo de espera entre llegadas.

Es(tau) = 1/lambda

mu > lambda: Restricción para que la longitud de la cola de peticiones no crezca indefinidamente.

tau: Intervalo entre llegadas.

Pn: Probabilidad de que haya "n" clientes en el sistema de colas en estado estable.

P0: Probabilidad de que el sistema se encuentre ocioso.

P0 = E^(-(lambda/mu))

Pri: Probabilidad de que hayan "i" peticiones pendientes.

Pri = ((lambda/mu)^i)(E^(-(lambda/mu))/Factorial[i])

(i=0,1,2,3,..)

Es(i): N° promedio de peticiones pendientes.

Es(i) = (lambda/mu) *)
```

```
BeginPackage["Colas2pn`"]
<<Statistics`LinearRegression`
Colas2pn::usage=
"Colas2pn[mu,lambda,i]
Análisis del rendimiento de un subsistema
de disco que puede dar servicio a más de
una petición a la vez.
Colocar los valores máximos para mu, lambda
y el número de peticiones pendientes."
Colas2pn[inmu_,inlambda_,ini_]:=
Colas2pnAux[inmu,inlambda,ini];
Colas2pnAux[inmu_,inlambda_,ini_]:=
Module[{coef, Pri, Es, listacoef, listapetpend,
result},
Caso2[mu_,lambda_,i_]:=
If[(N[lambda/mu] >= 1.),
Print["Para que la cola no
crezca indefinidamente debe
ser Mu mayor que Lambda."],
{coef=N[lambda/mu,6],
Pri=N[((lambda/mu)^i)*((E^(-(lambda/mu)))/(Factorial[i])),6],
Es=N[lambda/mu,6],
Print["Análisis del rendimiento de un"]
Print["subsistema de disco."]
Print["Resultados del Caso 2:"]
Print["El subsistema de disco "]
Print["puede dar servicio a varias"]
Print["peticiones a la vez."]
```

```
Print["Los valores de mu, lambda y el"]
Print["coeficiente son los siguientes:"]
Print[N[{mu,lambda,coef}]]
Print["La cola no crecerá"]
Print["indefinidamente debido a que"]
Print["Mu es mayor que Lambda."]
Print["Las probabilidades de tener"]
Print["0,1,2,3,4,...peticiones"]
Print["pendientes son las siguientes:"]
Print[N[Pri]]
Print["El promedio de peticiones"]
Print["pendientes es el siguiente:"]
Print[N[Es]]
Print["*****************************"]
AppendTo[listacoef,coef]
AppendTo[listapetpend,Es]
AppendTo[result,{coef,Es}]
}
];
Modelo2[mum_,lambdam_,im_]:=
If[(N[lambdam/mum] >= 1.),
Throw["Para que la cola no
crezca indefinidamente debe
ser Mu mayor que Lambda."],
For[i=10, i<=mum, i+=10,
For[j=1, j<=lambdam, j+=2,
Caso2[i,j,{Range[0,im,1]}]
];
```

```
];
];
listacoef={};
listapetpend={};
result={};
(* Colocar los valores máximos para mu, lambda *)
(* y el número de peticiones pendientes. *)
Modelo2[inmu,inlambda,ini];
Print[" "];
Print["****************************"];
Print["******* Resúmen final *******"];
Print["****************************"];
Print["Lista de coeficientes lambda/mu:"];
Print[listacoef];
Print["Promedio de peticiones"];
Print["pendientes para cada coeficiente:"];
Print[listapetpend];
Print["Pares (coeficientes,promedio):"];
Print[result];
Print["Gráfico de promedio de peticiones"];
Print["pendientes para cada coeficiente y"];
Print["Análisis de regresión lineal:"];
ListPlot[result,PlotStyle->
{RGBColor[1,0.6,0.3],PointSize[0.023]},
AxesLabel->{Coeficientes,Esperas},
AxesOrigin->{0,0}];
Regress[result,{1,x},x,
OutputList->{BestFit,
```

BestFitCoefficients,ANOVATable,

ConfidenceIntervalTable,

CovarianceMatrix,CorrelationMatrix,

EstimatedVariance,FitResiduals,

ParameterTable,PredictedResponse,

RSquared,AdjustedRSquared}]

]

EndPackage[];

18.6 Datos y Ejecuciones

Los datos para la simulación se introducen por teclado, siendo ellos los valores de μ, λ y el número de peticiones pendientes.

Los resultados detallados de las ejecuciones se muestran paso a paso en pantalla y pueden ser grabados en el mismo archivo de datos y ejecuciones, que en este caso ha sido el archivo **Colas2en.ma.**

El contenido del mencionado archivo luego de varias ejecuciones de evaluación puede consultarse en los Anexos; un detalle parcial es el siguiente, aclarándose que no se ha incluido en el contenido mostrado el análisis estadístico detallado:

<<Examples‘Colas2pn‘

? Colas2pn

Colas2pn[mu,lambda,i] Análisis del rendimiento de un subsistema de disco que puede dar servicio a más de una petición a la vez.

Colocar los valores máximos para mu, lambda y el número de peticiones pendientes.

Colas2pn[20,10,3]

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806}}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.740818, 0.222245, 0.0333368, 0.00333368}}

El promedio de peticiones pendientes es el siguiente:

0.3

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.606531, 0.303265, 0.0758163, 0.0126361}}

El promedio de peticiones pendientes es el siguiente:

0.5

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.496585, 0.34761, 0.121663, 0.0283881}}

El promedio de peticiones pendientes es el siguiente:

0.7

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.40657, 0.365913, 0.164661, 0.0493982}}

El promedio de peticiones pendientes es el siguiente:

0.9

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.951229, 0.0475615, 0.00118904, 0.0000198173}}

El promedio de peticiones pendientes es el siguiente:

0.05

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.860708, 0.129106, 0.00968296, 0.000484148}}

El promedio de peticiones pendientes es el siguiente:

0.15

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.778801, 0.1947, 0.0243375, 0.00202813}}

El promedio de peticiones pendientes es el siguiente:

0.25

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.704688, 0.246641, 0.0431621, 0.00503558}}

El promedio de peticiones pendientes es el siguiente:

0.35

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.637628, 0.286933, 0.0645599, 0.00968398}}

El promedio de peticiones pendientes es el siguiente:

0.45

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45}

Promedio de peticiones pendientes para cada coeficiente:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45}

Pares (coeficientes,promedio):

{{0.1, 0.1}, {0.3, 0.3}, {0.5, 0.5}, {0.7, 0.7}, {0.9, 0.9}, {0.05, 0.05},

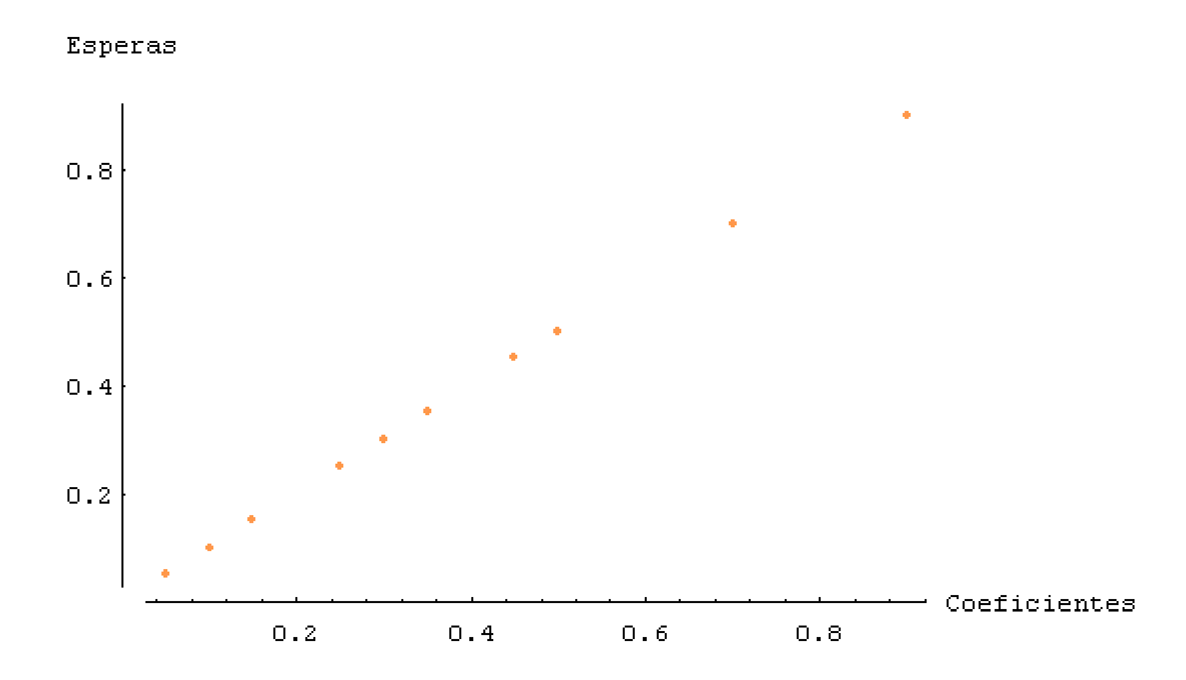


Figura 18.1: Promedio de peticiones pendientes.

{0.15, 0.15}, {0.25, 0.25}, {0.35, 0.35}, {0.45, 0.45}}

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

{Statistics‘LinearRegression‘BestFit ->

-2.37982 10^{-17} + 1. x,

Statistics‘LinearRegression‘BestFitCoefficients ->

{-2.37982 10^{-17} , 1.}}

18.7 Resultados y Conclusiones

La utilización de *Mathematica* para la resolución del problema planteado ha resultado muy satisfactoria, destacándose las facilidades y potencia del producto.

Los *resultados obtenidos ratifican*, como era de esperarse, las *previsiones teóricas* en cuanto a las *diferencias* en tiempos en cola de espera en disco de las distintas peticiones, según las distintas *cargas de trabajo* simuladas.

La modalidad implementada de mostrar los resultados paso a paso por pantalla permite observar el comportamiento del algoritmo y facilita la comprensión de su funcionamiento.

Como era de esperarse, se observa que en todos los casos, *la forma de la curva de los gráficos* obtenidos es similar, apreciándose un muy buen ajuste de los valores obtenidos (y graficados) a una recta.

Asimismo se observa un *leve impacto en los tiempos de espera* ante incrementos de la *carga de trabajo*, expresada según los coeficientes considerados, obteniéndose que el número promedio de peticiones pendientes tiende a 1.

Capítulo 19

Optimización de Operaciones de Búsqueda en Disco con Redes Neuronales

19.1 Introducción

Para el desarrollo del presente caso de estudio se ha utilizado el software **Mathematica,** una muy poderosa herramienta para el *cálculo numérico* y el especialmente para el *cálculo simbólico* [5, Castillo, Iglesias, Gutiérrez, Alvarez y Cobo], como así también software específico de *redes neuronales artificiales*, a saber: **Nndt, Nnmodel** y **Qnet** [**10, Hilera y Martínez**].

Las *operaciones de acceso a disco*, consecuencia de los requerimientos de lectura y/o grabación que efectúan al Sistema Operativo los distintos procesos, son la causa de considerables demoras en los tiempos de respuesta; dichas demoras son especialmente significativas si consideramos que las *operaciones de procesador* demandan tiempos de ciclo del orden del nanosegundo, en tanto que las operaciones de acceso a disco insumen tiempos del orden del milisegundo, es decir que la diferencia es de seis órdenes de magnitud.

Asimismo, si consideramos cómo se compone el *tiempo de acceso a disco*, encontramos que el mismo está integrado por el *tiempo de búsqueda* (movimiento del mecanismo de acceso que contiene el cabezal de lectura / grabación hasta posicionarse en el cilindro que contiene a la pista que aloja al registro que se debe acceder), el *tiempo de demora rotacional* o *latencia* (tiempo que demora la rotación del disco hasta que el registro que se pretende acceder queda bajo el cabezal de lectura / grabación, previamente posicionado en el cilindro que aloja a la pista correcta) y el *tiempo de transferencia* de la información propiamente dicha (lectura o grabación) [7, Deitel] y [23, Tanenbaum].

Además de lo indicado precedentemente se debe señalar que de los tres tiempos mencionados, el tiempo de búsqueda generalmente es el mayor en la generalidad de los dispositivos de disco, razón por la cual se ha tratado de disminuirlo permanentemente, habiéndose llegado (con *modificaciones de diseño* y de *tecnología de fabricación*), en algunos modelos de disco, a aproximarlo al tiempo de demora rotacional o latencia.

Otra vía para mejorar los tiempos de acceso a disco y por ende la performance de los procesos, consiste en *optimizar los algoritmos de búsqueda*, teniendo en cuenta que cuando

existe un conjunto de requerimientos de acceso sobre disco, dichos requerimientos son encolados por el Sistema Operativo y atendidos según un cierto *algoritmo de planificación*, el cual debe indicar el *orden* en que serán atendidas las distintas peticiones de acceso previamente encoladas.

Lo señalado precedentemente lleva a la conclusión de que dependiendo del *criterio de optimización* que se considere, el *orden* de atención de las distintas peticiones de acceso encoladas será *distinto*; no obstante, generalmente el criterio aplicado es el de lograr atender a las peticiones encoladas con el *mínimo de movimientos del mecanismo de acceso*, es decir con el mínimo de saltos de un cilindro a otro.

Esto nos lleva a la conveniencia de que los Sistemas Operativos puedan elegir ante cada conjunto de peticiones de acceso a disco, el *algoritmo más adecuado* a dicho conjunto, que permita atender todas las peticiones con el *mínimo de saltos de un cilindro a otro del disco;* el problema es que los Sistemas Operativos generalmente utilizan *un solo algoritmo* para cualquier conjunto posible de peticiones de acceso y no seleccionan el más adecuado para cada conjunto; una forma de solucionar este problema sería instrumentar una *Red Neuronal Artificial* en el entorno del Sistema Operativo o de los controladores de disco, para que con el criterio de minimizar el movimiento del mecanismo de acceso a disco, seleccione para cada conjunto de peticiones el algoritmo de búsqueda que se debería utilizar.

Los aspectos teóricos relacionados con los algoritmos de búsqueda han sido desarrollados en el Capítulo *"Entrada / Salida"*.

19.2 Objetivo del Caso de Estudio

Conforme a lo antes indicado, el objetivo del caso de estudio desarrollado fue el de *programar un paquete con Mathematica* (**Colas3pn.m**) que *implementara los principales algoritmos de búsqueda* y que permitiera *generar archivos para entrenamiento y testeo de redes neuronales* (**Colas3en.ma** y **Colas3fn.txt)** y posteriormente efectuar comparaciones entre los distintos modelos y herramientas de redes utilizados.

El *paquete desarrollado con Mathematica* permite analizar un conjunto de requerimientos con *tres algoritmos de búsqueda distintos,* calculando para cada uno de ellos la cantidad de movimientos del mecanismo de acceso (número de cilindros del disco recorridos por el mecanismo de acceso) y señalando en cada caso cuál es el más adecuado según el criterio indicado de *minimizar* los movimientos; además de desplegar los datos y resultados en pantalla, el paquete genera un archivo de texto que permite *entrenar y testear las redes neuronales*, teniendo presente que se trata de un *problema de clasificación.*

19.3 Descripción del Problema Planteado

Una descripción detallada del problema planteado puede consultarse en el Capítulo *"Entrada / Salida"*, siendo especialmente pertinentes los temas relacionados con *Algoritmos de Programación del Brazo del Disco* y *Optimización de la Búsqueda en Discos.*

19.4 Descripción de los Algoritmos Utilizados

Al igual que en el apartado anterior, una descripción detallada de los algoritmos aplicables a este caso, puede consultarse en el Capítulo *"Entrada / Salida"*, siendo especialmente pertinentes los temas sobre *Algoritmos de Programación del Brazo del Disco* y *Optimización de la Búsqueda en Discos* [7, Deitel].

Los *algoritmos* mencionados precedentemente pueden resumirse de la siguiente manera:

- **FCFS**: Las operaciones requeridas se atienden según el *orden de llegada* de los requerimientos.
- **SSF**: Las operaciones requeridas se atienden dando prioridad a las que significan *menos movimiento* (salto de cilindros) del mecanismo de acceso.
- **SCAN o del Elevador**: Las operaciones requeridas se atienden dando prioridad a las que significan *menos movimiento* del mecanismo de acceso, pero *respetando el sentido del movimiento* del mismo (desde los cilindros (conjuntos de pistas homónimas) de dirección más alta hacia los cilindros de dirección más baja, o viceversa).

En cuanto a las herramientas utilizadas para el entrenamiento y testeo de diferentes redes, las mismas fueron las siguientes: **Nndt, Nnmodel** y **Qnet**.

Las *etapas* cumplidas en el desarrollo del caso fueron las siguientes:

- Desarrollo del paquete de Mathematica para generar los archivos de entrenamiento y testeo de las distintas redes.
- Estudio y evaluación con la herramienta Nndt.
- Estudio y evaluación con la herramienta Nnmodel.
- Estudio y evaluación con la herramienta Qnet.
- Análisis final de los resultados y emisión de las conclusiones.

19.5 Programa Desarrollado

El paquete de Mathematica desarrollado se encuentra en el archivo **Colas3pn.m** y posee las siguientes características:

Se lo puede invocar utilizando, por ejemplo, el archivo **Colas3en.ma** que recibe como entradas los datos referidos a:

- Número de cilindros del dispositivo de disco que se simulará.
- Total de peticiones que se evaluarán en cada ciclo, es decir el número máximo de peticiones que se pueden encolar para ser luego atendidas según algún algoritmo de búsqueda.
- Número de ciclos que se evaluarán, es decir número de registros que tendrá el archivo generado.

Es necesario señalar además que las peticiones de acceso a determinadas pistas del disco se expresan por el número del cilindro al que pertenece la pista, siendo estos requerimientos generados (por el paquete) al azar, al igual que el número que determina la posición inicial del mecanismo de acceso antes de atender las peticiones de cada ciclo.

Respecto del archivo generado, el mismo tendrá un tamaño variable según los datos que se hayan suministrado al paquete que lo produce, siendo posible efectuar la generación del archivo en varias etapas, es decir en sucesivas ejecuciones del paquete, en cuyo caso y por una simple cuestión de estructura de los datos, el número de cilindros y el número de peticiones por ciclo deberán ser los mismos, ya que de lo contrario el archivo generado no podrá ser utilizado para el entrenamiento y testeo de las redes directamente por no ser homogéneo, sino que habría que segmentarlo con algún editor previamente.

El *código del programa desarrollado* (**Colas3pn.m**) es el siguiente:

```
(* ALGORITMOS DE BUSQUEDA EN DISCO *)

(* Cálculo del número de cilindros del disco

recorridos por el mecanismo de acceso en las

operaciones de búsqueda de los cilindros

requeridos según las peticiones de entrada /

salida que precisan los procesos; generación

del archivo de resultados correspondiente *)

(* Caso 1: Algoritmo FCFS: Las operaciones requeridas

se atienden según el orden de llegada de los

requerimientos. *)

(* Caso 2: Algoritmo SSF: Las operaciones requeridas

se atienden dando prioridad a las que

significan menos movimiento (salto de

cilindros) del mecanismo de acceso. *)

(* Caso 3: Algoritmo SCAN o del Elevador: Las operaciones

requeridas se atienden dando prioridad a las

que significan menos movimiento del mecanismo

de acceso, pero respetando el sentido del

movimiento del mismo (desde los cilindros

(conjuntos de pistas homónimas) de dirección
```

más alta hacia los cilindros de dirección
más baja, o viceversa. *)
(* Referencias y aclaraciones:
cildisc: Total de cilindros en el disco.
peticalea: Lista de cilindros involucrados
en las peticiones, según el orden
de llegada de las mismas.
totpetic: Total máximo de peticiones en
una lista de peticiones.
posinit: Posición inicial del mecanismo de acceso.
numpetic: Número de listas de peticiones.
tfcfs: Total de cilindros recorridos por
el mecanismo de acceso según
planificación FCFS.
tssf: Total de cilindros recorridos por
el mecanismo de acceso según
planificación SSF.
tscan: Total de cilindros recorridos por
el mecanismo de acceso según
planificación SCAN. *)
BeginPackage["Colas3pn`"]
Colas3pn::usage=
"Colas3pn[cildisc,totpetic,numpetic] \n
Cálculo del número de cilindros del disco \n
recorridos por el mecanismo de acceso en las \n
operaciones de búsqueda de los cilindros \n
requeridos según las peticiones de entrada / \n
salida que precisan los procesos; generación \n

```
del archivo de resultados correspondiente. \n
Colocar los valores para cildisc, totpetic y \n
numpetic.”
Colas3pn[incildisc_,intotpetic_,innumpetic_]:=
Colas3pnAux[incildisc,intotpetic,innumpetic];
Colas3pnAux[incildisc_,intotpetic_,innumpetic_]:=
Module[{tfcfs, tssf, tscan, fcfs, ssf, scan,
peticalea, posinit, result, resultclas,
peticiones, peticaux, movmin, indice,
posinitaux},
Caso3[cildisc_,totpetic_,numpetic_]:=
(posinit=Random[Integer, {1, cildisc}];
peticalea=Table[Random[Integer, {1, cildisc}],
{totpetic}];
tfcfs=0;
tssf=0;
tscan=0;
fcfs=0;
ssf=0;
scan=0;
indice=0;
(* Cálculo para FCFS *)
tfcfs=Abs[peticalea[[1]]-posinit];
For[j=2, j<=totpetic, j+=1,
tfcfs=tfcfs+Abs[peticalea[[j]]-peticalea[[j-1]]];
];
(* Cálculo para SSF *)
peticiones=peticalea;
```

```
posinitaux=posinit;
For[k=1, k<=totpetic, k+=1,
movmin=200000;
For[j=1, j<=totpetic, j+=1,
peticaux=Abs[posinitaux-peticiones];
If[peticaux[[j]]<movmin, movmin=peticaux[[j]];
indice=j];
];
tssf=tssf+movmin;
peticiones[[indice]]=10000;
posinitaux=peticalea[[indice]];
];
(* Cálculo para SCAN *)
peticiones=peticalea;
For[j=1, j<=totpetic, j+=1,
If[peticiones[[j]]<=posinit, peticiones[[j]]=posinit;];
];
peticiones=Sort[peticiones];
tscan=Abs[peticiones[[1]]-posinit];
For[j=2, j<=totpetic, j+=1,
tscan=tscan+Abs[peticiones[[j]]-peticiones[[j-1]]];
];
posinitaux=peticiones[[totpetic]];
peticiones=peticalea;
For[j=1, j<=totpetic, j+=1,
If[peticiones[[j]]>=posinit, peticiones[[j]]=posinitaux;];
];
peticiones=Reverse[Sort[peticiones]];
```

```
tscan=tscan+Abs[peticiones[[1]]-posinitaux];
For[j=2, j<=totpetic, j+=1,
tscan=tscan+Abs[peticiones[[j]]-peticiones[[j-1]]];
];
result={tfcfs,tssf,tscan};
resultclas=Sort[result];
If[tfcfs==resultclas[[1]], fcfs=1,
If[tssf==resultclas[[1]], ssf=1,
If[tscan==resultclas[[1]], scan=1
];
];
];
Print["Cálculo de movimientos del "];
Print["mecanismo de acceso según tres "];
Print["algoritmos de búsqueda."];
Print["Los valores de cildisc, totpetic y "];
Print["numpetic son los siguientes:"];
Print[{cildisc,totpetic,numpetic}];
Print["La posición inicial del mecanismo "];
Print["de acceso es la siguiente:"];
Print[posinit];
Print["La lista aleatoria de peticiones "];
Print["es la siguiente:"];
Print[peticalea];
Print["El número de movimientos del "];
Print["mecanismo de acceso sería el "];
Print["siguiente para FCFS, SSF y SCAN:"];
Print[result];
```

```
Print["El algoritmo más eficiente es:"];
If[fcfs==1, Print["FCFS"],
If[ssf==1, Print["SSF"],
If[scan==1, Print["SCAN"]
];
];
];
Print["Imagen del registro grabado: "];
Print[{posinit,peticalea,result,fcfs,ssf,scan}];
PutAppend[{posinit,peticalea,fcfs,ssf,scan},
"D:\Wnmath22\Trabajo\Colas3fn.txt"];
Print["*****************************"];
);
Modelo3[cildiscm_,totpeticm_,numpeticm_]:=
For[i=1, i<=numpeticm, i+=1,
Caso3[cildiscm,totpeticm,i]
];
peticalea={};
peticiones={};
peticaux={};
result={};
resultclas={};
(* Colocar los valores para número de cilindros, *)
(* total de peticiones que se evalúan en cada ciclo y *)
(* número de ciclos que se evaluarán, es decir *)
(* número de registros que tendrá el archivo generado. *)
Print["Se sugiere que numpetic sea al menos "];
Print["30 veces mayor que totpetic."];
```

```
Print[" "];
(* Put[{}, "D:\Wnmath22\Trabajo\Colas3fn.txt"]; *)
Modelo3[incildisc,intotpetic,innumpetic];
Print["*********************************************"];
Print["** Archivo de datos grabado como Colas3fn.txt **"];
Print["*********************************************"];
]
EndPackage[];
```

19.6 Datos y Ejecuciones

Los resultados detallados de las ejecuciones se muestran paso a paso en pantalla y pueden ser grabados en el mismo archivo de datos y ejecuciones, que en este caso ha sido el archivo **Colas3en.ma,** en tanto que los datos resumen para el entrenamiento y testeo de las redes neuronales, como ya se indicó, se graban en el archivo **Colas3fn.txt**.

El archivo generado (**Colas3fn.txt)**, posee la siguiente estructura:

- Número del cilindro que corresponde a la posición inicial del mecanismo de acceso.
- Lista de peticiones atendidas en un ciclo, las que están representadas por el número del cilindro que aloja en cada caso a la pista que debe ser accedida (interesa el cilindro y no la pista por cuanto acceder a una pista situada en el mismo cilindro que otra no significa un movimiento de posicionamiento del mecanismo de acceso, es decir no significa un salto de cilindro); además se debe tener presente que el número de peticiones atendidas en un ciclo es un dato de entrada, en tanto que la lista de peticiones de cada ciclo se genera al azar.
- Calificación obtenida por el método de búsqueda (algoritmo) FCFS, donde cero (0) significa no seleccionado y uno (1) significa seleccionado.
- Calificación obtenida por el método de búsqueda (algoritmo) SSF, donde cero (0) significa no seleccionado y uno (1) significa seleccionado.
- Calificación obtenida por el método de búsqueda (algoritmo) SCAN, donde cero (0) significa no seleccionado y uno (1) significa seleccionado.

Atento a que el archivo generado por el paquete (**Colas3fn.txt)** incluye caracteres (llaves) que lo hacen inadecuado para su utilización directa en las herramientas de redes neuronales, se ha procedido a generar un archivo similar sin dichos caracteres conflictivos utilizando un editor (puede ser cualquier editor, incluso el del propio Mathematica); el archivo mencionado precedentemente es el **Colas3fn.ma**, que finalmente dio lugar al archivo **Colas3fn.dat**. Esto significa que en todos los casos la configuración de red utilizada posee seis (6) neuronas de entrada y tres (3) de salida.

El contenido del mencionado archivo **Colas3en.ma** luego de una de las ejecuciones de evaluación puede consultarse en los Anexos; un detalle parcial es el siguiente:

<<Examples‘Colas3pn‘

? Colas3pn

Colas3pn[cildisc,totpetic,numpetic]

Cálculo del número de cilindros del disco recorridos por el mecanismo de acceso en las operaciones de búsqueda de los cilindros requeridos según las peticiones de entrada / salida que precisan los procesos; generación del archivo de resultados correspondiente.

Colocar los valores para cildisc, totpetic y numpetic.

Colas3pn[40,5,100]

Se sugiere que numpetic sea al menos 30 veces mayor que totpetic.

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 1}

La posición inicial del mecanismo de acceso es la siguiente:

23

La lista aleatoria de peticiones es la siguiente:

{16, 13, 9, 31, 39}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{44, 44, 46}

El algoritmo más eficiente es:

FCFS

Imagen del registro grabado:

{23, {16, 13, 9, 31, 39}, {44, 44, 46}, 1, 0, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 2}

La posición inicial del mecanismo de acceso es la siguiente:

14

La lista aleatoria de peticiones es la siguiente:

{29, 13, 28, 26, 39}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{61, 27, 51}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{14, {29, 13, 28, 26, 39}, {61, 27, 51}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 3}

La posición inicial del mecanismo de acceso es la siguiente:

21

La lista aleatoria de peticiones es la siguiente:

{6, 38, 23, 4, 31}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{108, 51, 51}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{21, {6, 38, 23, 4, 31}, {108, 51, 51}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 4}

La posición inicial del mecanismo de acceso es la siguiente:

5

La lista aleatoria de peticiones es la siguiente:

{34, 13, 15, 20, 30}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{67, 29, 29}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{5, {34, 13, 15, 20, 30}, {67, 29, 29}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 5}

La posición inicial del mecanismo de acceso es la siguiente:

32

La lista aleatoria de peticiones es la siguiente:

{21, 28, 33, 12, 31}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{63, 22, 22}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{32, {21, 28, 33, 12, 31}, {63, 22, 22}, 0, 1, 0}

****

****

** Archivo de datos grabado como Colas3fn.txt **

****

El contenido del mencionado archivo **Colas3fn.dat** luego de una de las ejecuciones de evaluación puede consultarse en los Anexos; un detalle parcial es el siguiente:

14, 15, 2, 20, 13, 7 , 0, 0, 1

37, 9, 11, 23, 32, 36 , 0, 1, 0

6, 30, 31, 15, 26, 11 , 0, 1, 0

11, 25, 38, 34, 24, 26 , 0, 1, 0

5, 24, 25, 27, 32, 9 , 0, 1, 0

31, 18, 32, 4, 35, 20 , 0, 1, 0

38, 20, 27, 4, 24, 22 , 0, 1, 0

39, 27, 4, 30, 6, 27 , 0, 1, 0

12, 28, 33, 2, 40, 17 , 0, 1, 0

12, 24, 19, 12, 15, 19 , 0, 1, 0

40, 4, 6, 24, 20, 24 , 0, 1, 0

7, 7, 17, 32, 13, 25 , 0, 1, 0

29, 16, 2, 38, 36, 5 , 0, 1, 0

8, 1, 15, 11, 24, 23 , 1, 0, 0

38, 17, 22, 40, 11, 39 , 0, 1, 0

26, 4, 34, 2, 21, 4 , 0, 0, 1

27, 30, 3, 22, 38, 35 , 0, 1, 0

32, 13, 22, 2, 16, 4 , 0, 1, 0

4, 16, 6, 16, 26, 19 , 0, 1, 0

4, 18, 19, 2, 36, 34 , 0, 1, 0

19.7 Descripción del Software de RNA Utilizado

19.7.1 Breve Introducción a las RNA

Las **RNA** o *redes neuronales artificiales* constituyen un importante paradigma en el contexto de la *inteligencia artificial [17, Castillo, Cobo, Gutiérrez y Pruneda], [9, Castillo, Gutiérrez y Hadi].* El elemento clave de este paradigma es una estructura computacional compuesta de un gran número de pequeños elementos procesadores interconectados *(neuronas)* trabajando en paralelo.

Las RNA constituyen estructuras de computación alternativas, creadas con el propósito de "reproducir" las funciones del cerebro humano; frecuentemente se hace referencia a dichas estructuras como *"computación neuronal".*

Las RNA están formadas por un gran número de procesadores, o neuronas, dispuestos en varias capas e interconectados entre sí mediante conexiones con pesos. Estos procesadores realizan cálculos simples basados en la información que reciben de los procesadores vecinos y utilizan un proceso de *aprendizaje* por analogía donde los pesos de las conexiones son ajustados automáticamente para reproducir un conjunto de patrones representativos del problema a aprender.

Según *[17, Castillo, Cobo, Gutiérrez y Pruneda]* una *neurona* o *unidad procesadora,* sobre un conjunto de nodos N, es una tripleta (X, f, Y), donde X es un subconjunto de N, Y es un único nodo de N y $f{:}I\!R \rightarrow I\!R$ es una *función neuronal* (también llamada *función de activación*) que calcula un valor de salida para Y basado en una combinación lineal de los valores de las componentes de X, es decir, $Y = f\left(\sum_{x_i \epsilon X} w_i\ x_i\right)$.

Los elementos X, Y y f se denominan conjunto de nodos de entrada, nodo de salida y función neuronal de la unidad neuronal, respectivamente.

También según *[17, Castillo, Cobo, Gutiérrez y Pruneda]* una *red neuronal artificial (RNA)* es un par (N, U), donde N es un conjunto de nodos y U es un conjunto de unidades procesadoras sobre N que satisface la siguiente condición: Cada nodo $X_i \in N$ tiene que ser un nodo de entrada o de salida de al menos una unidad procesadora de U.

Según Kohonen las *redes neuronales artificiales* son *redes interconectadas masivamente en paralelo* de elementos simples (usualmente adaptativos) y con *organización jerárquica*, las cuales intentan *interactuar con los objetos del mundo real* del mismo modo que lo hace el *sistema nervioso biológico [10, Hilera y Martínez].*

Las *funciones de activación* más utilizadas son las siguientes *[17, Castillo, Cobo, Gutiérrez y Pruneda]*:

- Funciones lineales.
- Funciones escalón.
- Funciones sigmoidales.

Las neuronas se pueden organizar en capas conectadas por varios *tipos de uniones*, a saber:

- Conexiones hacia adelante.
- Conexiones laterales.

- Conexiones hacia atrás o recurrentes.

Una unidad se dice que está en la *capa de entrada* de una red neuronal *(X, U)*, si es la entrada de al menos una unidad procesadora de *U* y no es la salida de ninguna unidad procesadora de *U*.

Una unidad se dice que está en la *capa de salida* de una red neuronal *(X, U)*, si es la salida de al menos una unidad procesadora de *U* y no es la entrada de ninguna unidad procesadora de *U*.

Una unidad se dice que está en la *capa intermedia u oculta* de una red neuronal *(X, U)*, si es la entrada de al menos una unidad procesadora de *U* y, al mismo tiempo, es la salida de al menos una unidad procesadora de *U*.

Una de las principales propiedades de las RNA es su capacidad de *aprender* a partir de unos datos. Una vez que ha sido elegida la *arquitectura de red* para un problema particular, los *pesos de las conexiones* se ajustan para codificar la información contenida en un conjunto de *datos de entrenamiento*.

Los principales *métodos de aprendizaje* son los siguientes *[17, Castillo, Cobo, Gutiérrez y Pruneda]*:

- Aprendizaje supervisado.
- Aprendizaje no supervisado:
  - Aprendizaje Hebbiano.
  - Aprendizaje competitivo.
  - Representación de características (Feature mapping).

Finalizado el proceso de aprendizaje de la red, con lo cual los pesos de la misma han sido calculados, se debe comprobar la calidad del modelo resultante *(validación)*, midiendo los errores entre los valores de salida deseados y los obtenidos por la red neuronal; algunas *medidas estándard del error* son:

- **SSE**: Sum of Square Errors: suma de los cuadrados de los errores.
- **RMSE**: Root Mean Square Error: raíz cuadrada del error cuadrático medio.
- Error máximo.

Las principales *ventajas* de la **RNA** son las siguientes [10, Hilera y Martínez]:

- Aprendizaje adaptativo.
- Autoorganización.
- Tolerancia a fallos.
- Operación en tiempo real.
- Fácil inserción dentro de la tecnología existente.

Algunas de las *posibles aplicaciones* de las **RNA** están en el ámbito de las siguientes disciplinas [10, Hilera y Martínez]:

- Biología.
- Empresa.
- Medio ambiente.
- Finanzas.
- Manufacturación.
- Medicina.
- Militares.

Por último y a los efectos de situar a las **RNA** dentro del mundo de la **computación** en general y frente a la **IA** (Inteligencia Artificial) en particular, ver la Tabla 19.1 de la página 511 [10, Hilera y Martínez].

	Computación *Convencional*	**Computación** *Simbólica*	**Computación** *Neuronal*
Basado en:	Arquitectura Von Neumann	Lógica Cognitiva	Neurobiología
Apropiada para:	Algoritmos conocidos	Heurística	Adaptación
No apropiada para:	Condiciones *difusas*	Causalidad desconocida	Cálculos precisos
Memoria:	Precisa, estática	Bases de conocimiento	Distribuida
Construida mediante:	Diseño, programación y prueba	Representación del conocimiento + motor de inferencia	Configuración y *aprendizaje*
Soporte:	Ordenadores secuenciales	Máquinas LISP	Procesadores paralelos

Tabla 19.1: Formas básicas de computación.

19.7.2 Herramienta Nndt

El software **Nndt** *(Neural Network Development Tool)* es una facilidad para el entrenamiento de redes neuronales artificiales.

La interface de usuario fue desarrollada con MS Visual Basic, Edición Profesional.

Las rutinas DLL (escritas en “C”) son usadas para muchas de las funciones matemáticas.

Los algoritmos de red implementados son de los llamados de tipo supervisado.

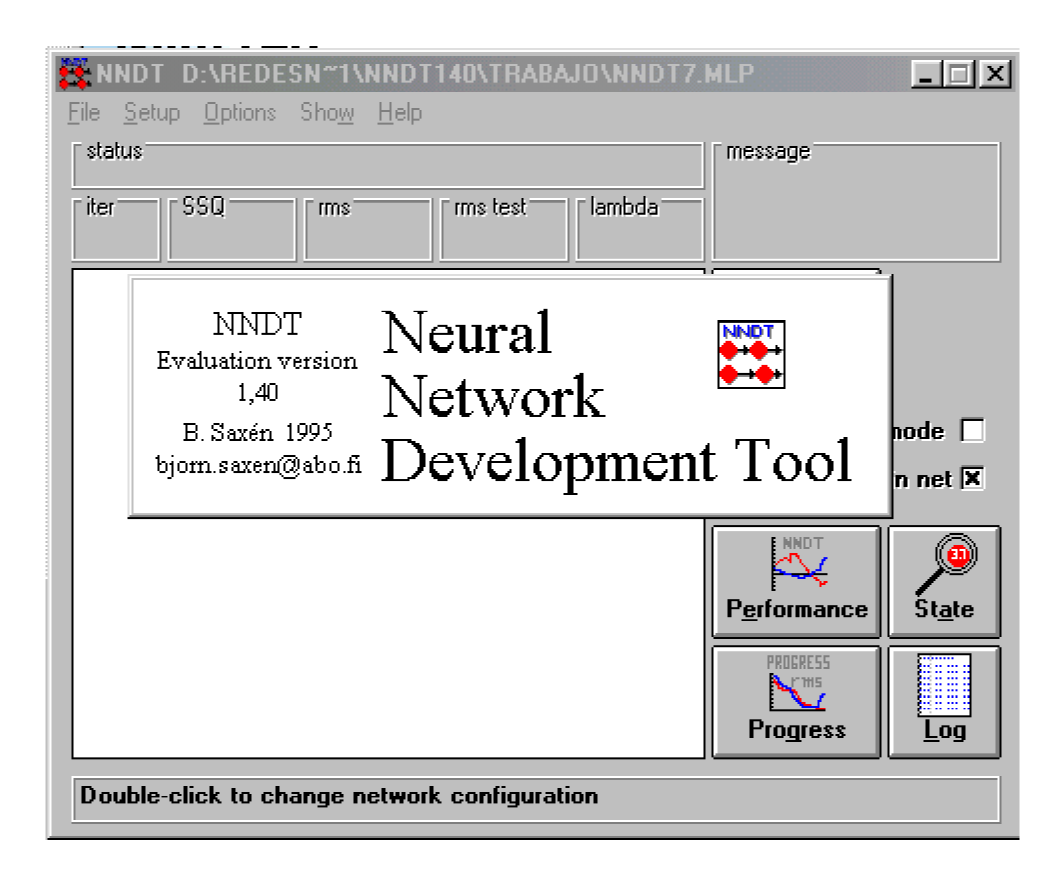


Figura 19.1: Ejemplo de pantalla Nndt.

Además, el producto incluye algoritmos para *redes perceptrón multicapa* (**MLP**: multi-layer perceptron) de los tipos feed-forward y recurrente.

La red MLP es entrenada con el *método Levenberg - Marquardt.*

El entrenamiento requiere un grupo de señales de entrada y las correspondientes de salida, almacenadas en un archivo referenciado como *archivo de pattern.* Este es el único archivo que el usuario debe suministrar. Opcionalmente, ciertos parámetros que definen las columnas del archivo de pattern, el tamaño de la red y la configuración de la red, pueden ser almacenados en un archivo referido como *archivo de setup.*

Nndt también incluye una rutina para presentación gráfica de señales de salida, activaciones de nodos, residuos y pesos durante la ejecución. La interface además provee facilidades para el examen de activaciones de nodos y pesos así como modificación de pesos.

Asimismo se incluye un archivo de help (ayuda) que puede consultarse en cualquier momento de la ejecución de **Nndt** presionando F1.

Algunas de las muchas pantallas del producto pueden observarse en la Figura 19.1 de la página 512 y en la Figura 19.2 de la página 513.

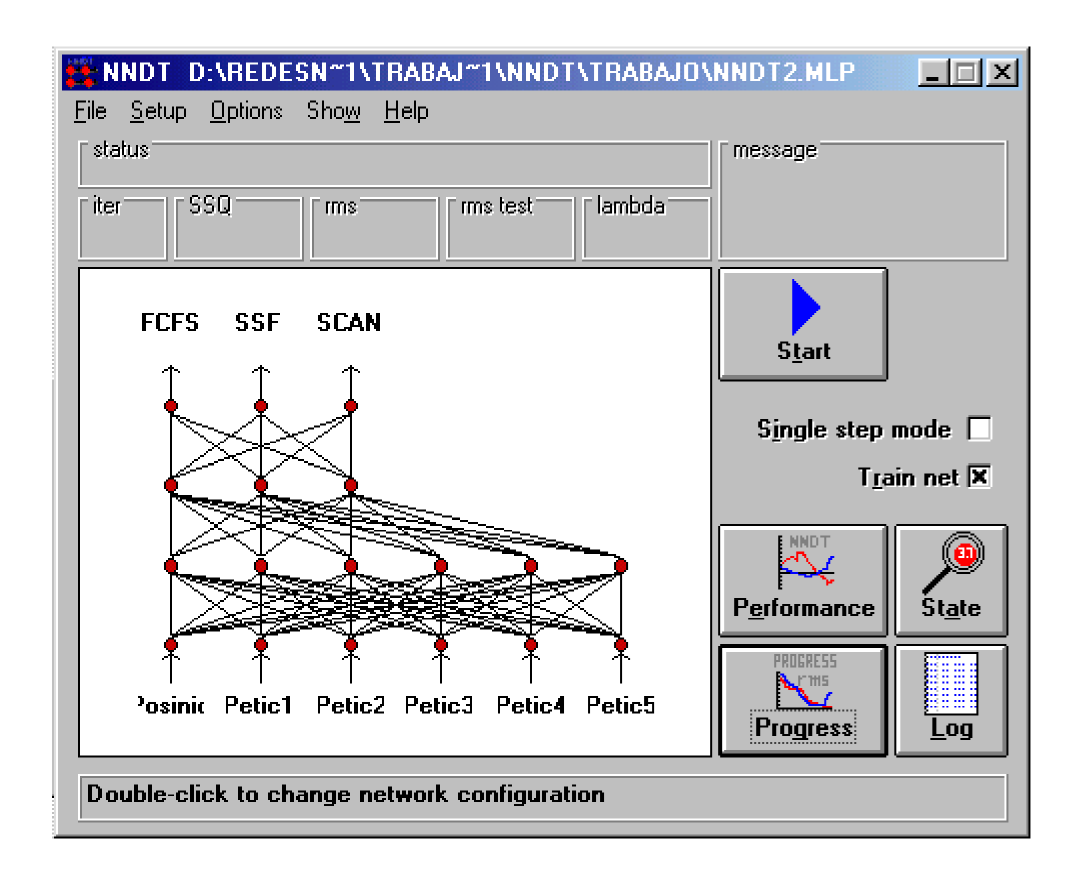


Figura 19.2: Ejemplo de pantalla Nndt.

19.7.3 Herramienta Nnmodel

Nnmodel es una herramienta para el modelado del proceso de datos, estadísticas de experimentos o bases de datos históricas. Puede buscar desde relaciones lineales simples hasta relaciones complejas no-lineales en datos empíricos. Es fácil de usar porque construye automáticamente modelos matemáticos directamente desde los datos del usuario. Permite crear modelos de prototipos rápida y fácilmente.

Nnmodel está diseñado para ayudar a obtener el máximo beneficio de las técnicas de modelado de poderosas redes neuronales sin requerir del usuario que aprenda un paquete de software complicado o un lenguaje estadístico.

Nnmodel contiene módulos para, entre otras, las siguientes *principales funciones:*

- Diseñar un experimento estadístico:
  - Permite crear una matriz de datos basada en un experimento diseñado estadísticamente.
- Ingreso de datos por teclado, archivo o clipboard:
  - Soporta tres métodos para el ingreso de los datos:
    1. Ingresar los datos directamente utilizando el editor de la matriz de datos.
    2. Importar una tabla ASCII o archivo delimitado con blancos.
    3. Pegar datos desde el clipboard de Windows.
- Ejecutar reportes estadísticos simples y correlaciones:
  - Permite generar un reporte con estadísticas básicas, tales como número de observaciones, máximo, mínimo, promedio, desviación estándard y suma de cuadrados.
  - También permite generar reportes de correlación conteniendo los coeficientes de correlación de Pearson e información complementaria de interés.
- Análisis gráfico de datos:
  - Permite visualizar gráficamente los datos utilizando una variedad de rutinas de plotting que incluyen, entre otras posibilidades, las siguientes:
    * Gráfico de tendencias por observación.
    * XY.
    * Distribución de frecuencias.
    * Visualización tridimensional.
- Manejo de datos históricos.
- Modelado de datos utilizando redes neuronales:
  - Permite que un modelo pueda ser creado y entrenado en pocos minutos.

- Interrogar al modelo interactivamente:
  - Luego de que el modelo ha sido entrenado es posible interrogar al modelo inmediatamente.
- Analizar la performance del modelo estadísticamente:
  - La performance del modelo puede ser evaluada utilizando el estadístico estándard R^2.
- Desplegar las predicciones del modelo gráficamente incluyendo gráficos 3D:
  - Una variedad de gráficos están disponibles, entre otros, los siguientes:
    - Datos vs. predicciones.
    - Residuos.
    - Tendencias.
    - Distribución de frecuencias.
    - XY.
    - Superficies 3D.
- Testear el modelo con un grupo de datos externos:
  - Una matriz de testeo puede ser cargada desde datos no originalmente utilizados para generar el modelo.
- Realizar análisis de sensitividad:
  - Este análisis puede mostrar cuán sensitiva es una variable de salida si se efectúan cambios en la entrada.
- Exportar el modelo neuronal como un archivo ASCII transportable:
  - El modelo de entrenamiento puede ser exportado desde Nnmodel a otra plataforma de hardware.
- DDE Interface:
  - Permite al usuario llamar a modelos pre-entrenados con cualquier programa que permite DDE (Dynamic Data Exchange).

Algunas de las muchas pantallas del producto pueden observarse en la Figura 19.3 de la página 516, en la Figura 19.4 de la página 517, en la Figura 19.5 de la página 518, en la Figura 19.6 de la página 519 y en la Figura 19.7 de la página 520.

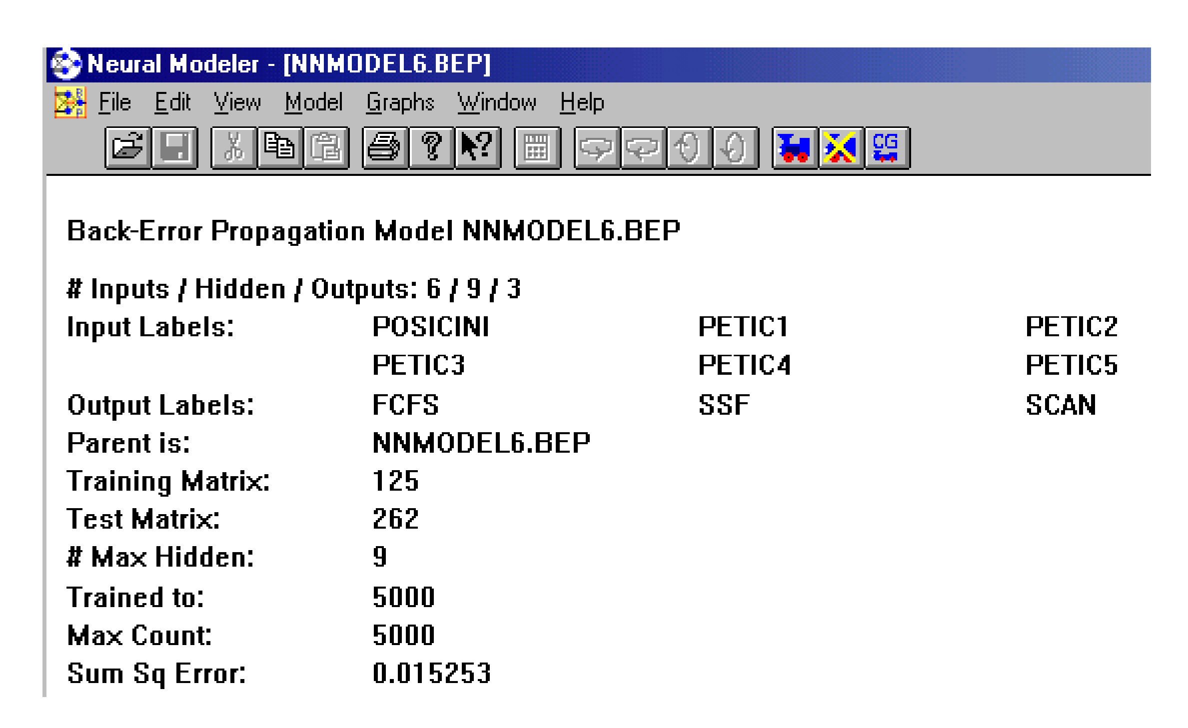


Figura 19.3: Ejemplo de pantalla Nnmodel.

19.7.4 Herramienta Qnet

Qnet es un poderoso simulador de redes neuronales artificiales, pero también fácil de utilizar.

Qnet implementa varios diferentes algoritmos para entrenar una red neuronal estándar *feed-forward* en un ambiente gráfico fácil de usar.

La versión shareware difiere de la registrada en el número de capas ocultas, el número de neuronas en las capas ocultas y el número de patterns por archivo de datos de entrenamiento.

Algunas de las muchas pantallas del producto pueden observarse en la Figura 19.8 de la página 521, en la Figura 19.9 de la página 522 y en la Figura 19.10 de la página 522.

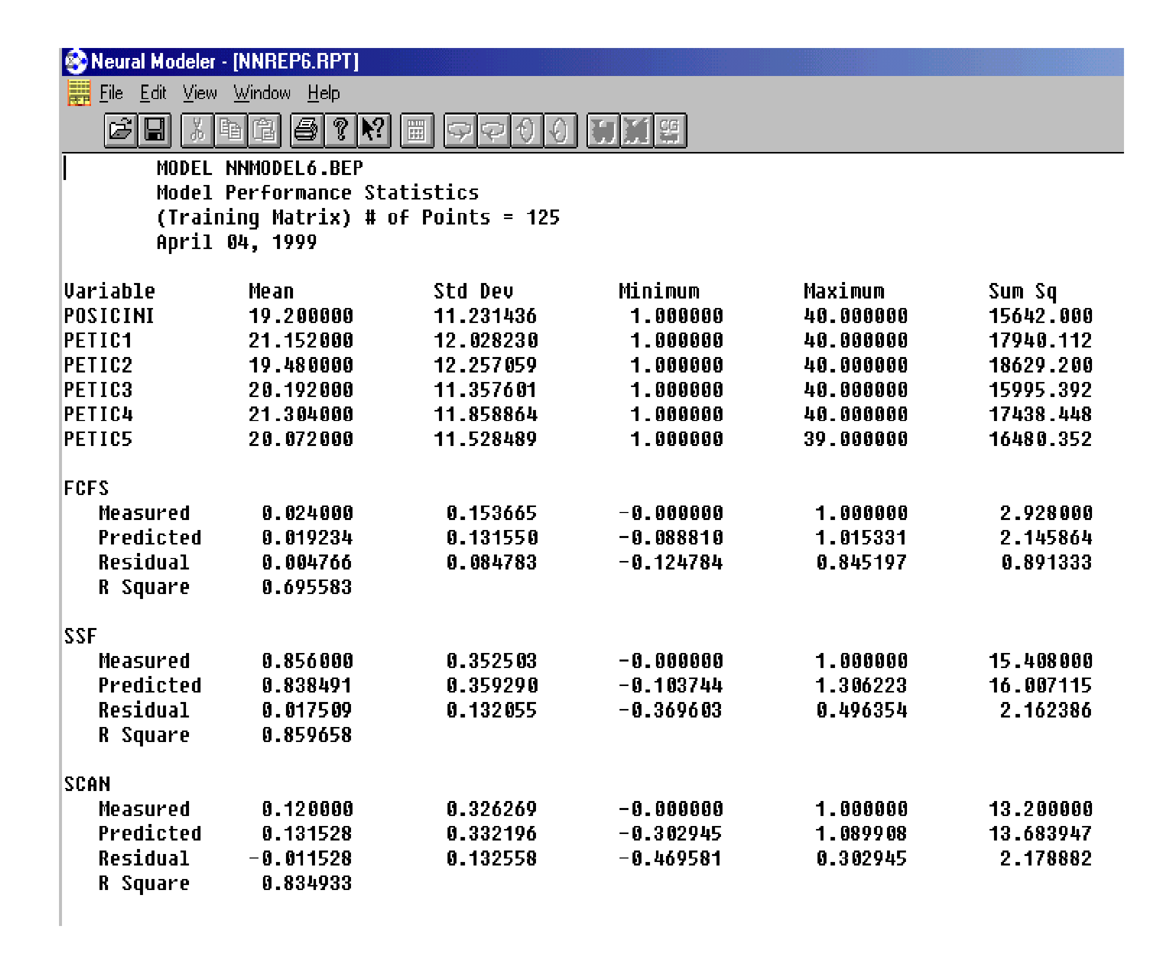
19.8 Resultados y Conclusiones

La utilización de *Mathematica* para la resolución del problema planteado ha resultado muy satisfactoria, destacándose las facilidades y potencia del producto.

Los *resultados obtenidos ratifican*, como era de esperarse, las *previsiones teóricas* en cuanto a las *diferencias* entre los distintos algoritmos de búsqueda respecto del *número de movimientos del mecanismo de acceso* requeridos para atender las mismas peticiones.

La modalidad implementada de mostrar los resultados paso a paso por pantalla permite observar el comportamiento de los algoritmos y facilita la comprensión de su funcionamiento.

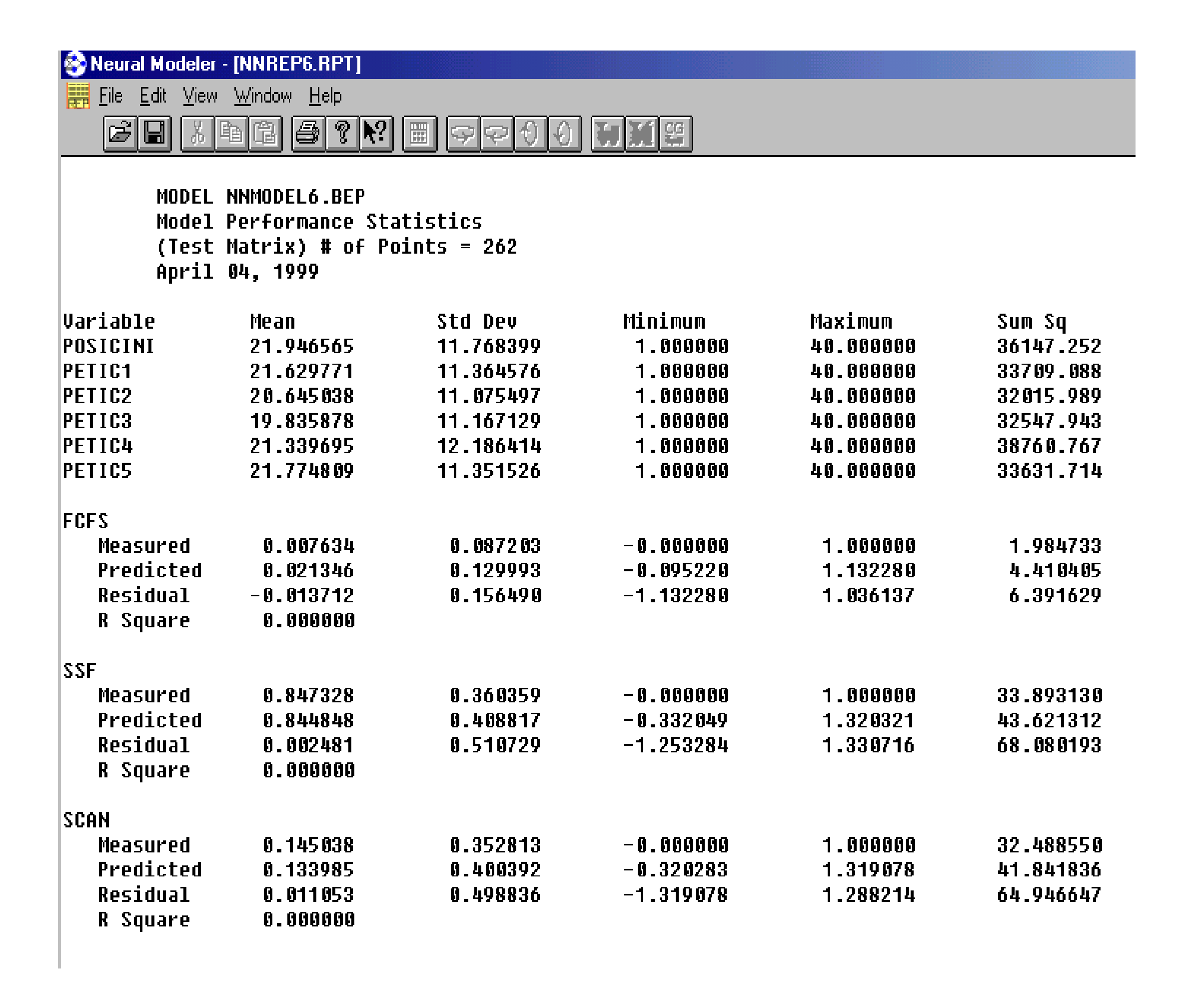
En lo referente a las herramientas de *redes neuronales artificiales* puede indicarse lo siguiente:



MODEL NNMODEL6.BEP
Model Performance Statistics
(Training Matrix) # of Points = 125
April 04, 1999

Variable	Mean	Std Dev	Minimum	Maximum	Sum Sq
POSICINI	19.200000	11.231436	1.000000	40.000000	15642.000
PETIC1	21.152000	12.028230	1.000000	40.000000	17940.112
PETIC2	19.480000	12.257059	1.000000	40.000000	18629.200
PETIC3	20.192000	11.357601	1.000000	40.000000	15995.392
PETIC4	21.304000	11.858864	1.000000	40.000000	17438.448
PETIC5	20.072000	11.528489	1.000000	39.000000	16480.352
FCFS					
Measured	0.024000	0.153665	-0.000000	1.000000	2.928000
Predicted	0.019234	0.131550	-0.088810	1.015331	2.145864
Residual	0.004766	0.084783	-0.124784	0.845197	0.891333
R Square	0.695583				
SSF					
Measured	0.856000	0.352503	-0.000000	1.000000	15.408000
Predicted	0.838491	0.359290	-0.103744	1.306223	16.007115
Residual	0.017509	0.132055	-0.369603	0.496354	2.162386
R Square	0.859658				
SCAN					
Measured	0.120000	0.326269	-0.000000	1.000000	13.200000
Predicted	0.131528	0.332196	-0.302945	1.089908	13.683947
Residual	-0.011528	0.132558	-0.469581	0.302945	2.178882
R Square	0.834933				

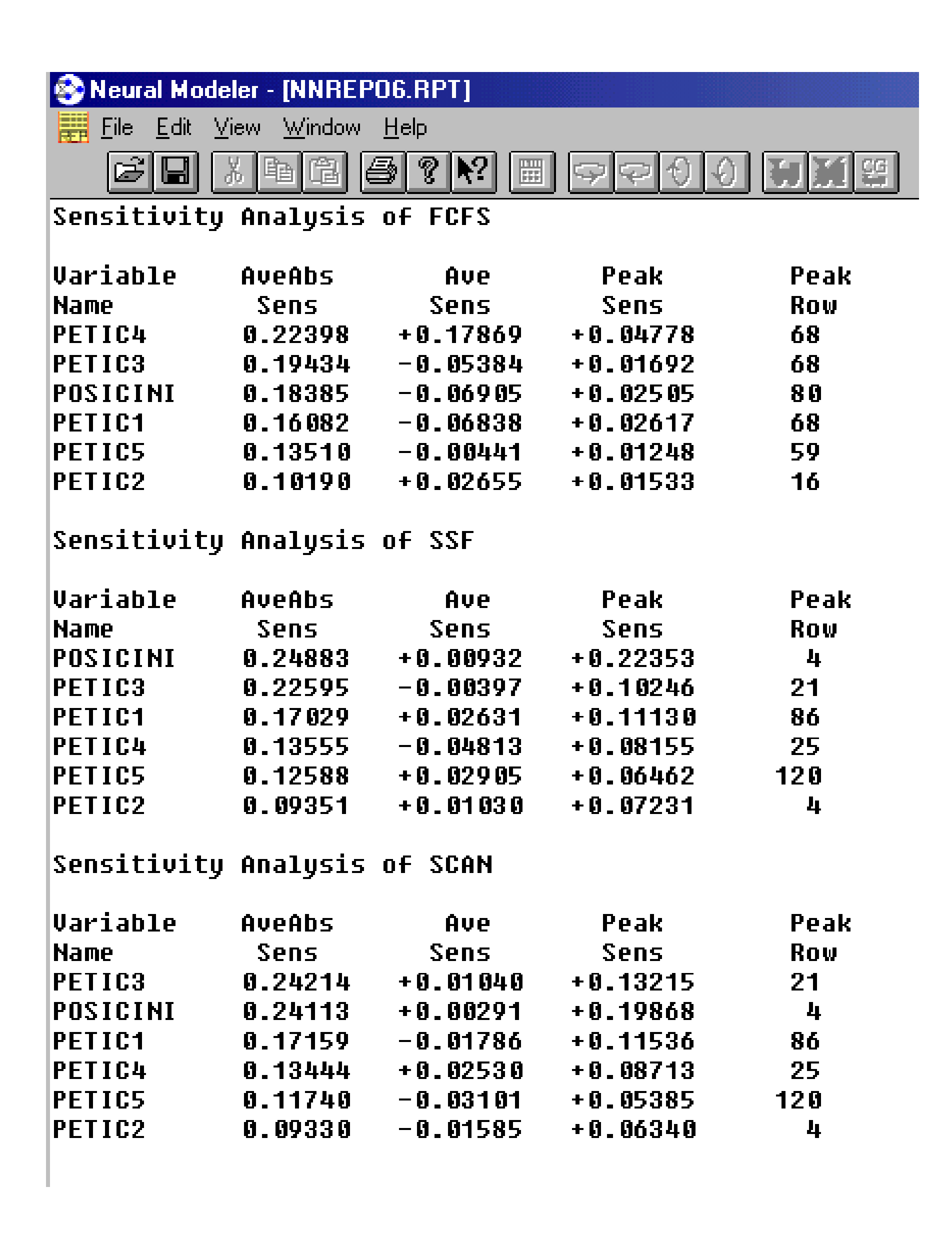
Figura 19.4: Ejemplo de pantalla Nnmodel.



MODEL NNMODEL6.BEP
Model Performance Statistics
(Test Matrix) # of Points = 262
April 04, 1999

Variable	Mean	Std Dev	Minimum	Maximum	Sum Sq
POSICINI	21.946565	11.768399	1.000000	40.000000	36147.252
PETIC1	21.629771	11.364576	1.000000	40.000000	33709.088
PETIC2	20.645038	11.075497	1.000000	40.000000	32015.989
PETIC3	19.835878	11.167129	1.000000	40.000000	32547.943
PETIC4	21.339695	12.186414	1.000000	40.000000	38760.767
PETIC5	21.774809	11.351526	1.000000	40.000000	33631.714
FCFS					
Measured	0.007634	0.087203	-0.000000	1.000000	1.984733
Predicted	0.021346	0.129993	-0.095220	1.132280	4.410405
Residual	-0.013712	0.156490	-1.132280	1.036137	6.391629
R Square	0.000000				
SSF					
Measured	0.847328	0.360359	-0.000000	1.000000	33.893130
Predicted	0.844848	0.408817	-0.332049	1.320321	43.621312
Residual	0.002481	0.510729	-1.253284	1.330716	68.080193
R Square	0.000000				
SCAN					
Measured	0.145038	0.352813	-0.000000	1.000000	32.488550
Predicted	0.133985	0.400392	-0.320283	1.319078	41.841836
Residual	0.011053	0.498836	-1.319078	1.288214	64.946647
R Square	0.000000				

Figura 19.5: Ejemplo de pantalla Nnmodel.



Sensitivity Analysis of FCFS

Variable Name	AveAbs Sens	Ave Sens	Peak Sens	Peak Row
PETIC4	0.22398	+0.17869	+0.04778	68
PETIC3	0.19434	-0.05384	+0.01692	68
POSICINI	0.18385	-0.06905	+0.02505	80
PETIC1	0.16082	-0.06838	+0.02617	68
PETIC5	0.13510	-0.00441	+0.01248	59
PETIC2	0.10190	+0.02655	+0.01533	16

Sensitivity Analysis of SSF

Variable Name	AveAbs Sens	Ave Sens	Peak Sens	Peak Row
POSICINI	0.24883	+0.00932	+0.22353	4
PETIC3	0.22595	-0.00397	+0.10246	21
PETIC1	0.17029	+0.02631	+0.11130	86
PETIC4	0.13555	-0.04813	+0.08155	25
PETIC5	0.12588	+0.02905	+0.06462	120
PETIC2	0.09351	+0.01030	+0.07231	4

Sensitivity Analysis of SCAN

Variable Name	AveAbs Sens	Ave Sens	Peak Sens	Peak Row
PETIC3	0.24214	+0.01040	+0.13215	21
POSICINI	0.24113	+0.00291	+0.19868	4
PETIC1	0.17159	-0.01786	+0.11536	86
PETIC4	0.13444	+0.02530	+0.08713	25
PETIC5	0.11740	-0.03101	+0.05385	120
PETIC2	0.09330	-0.01585	+0.06340	4

Figura 19.6: Ejemplo de pantalla Nnmodel.

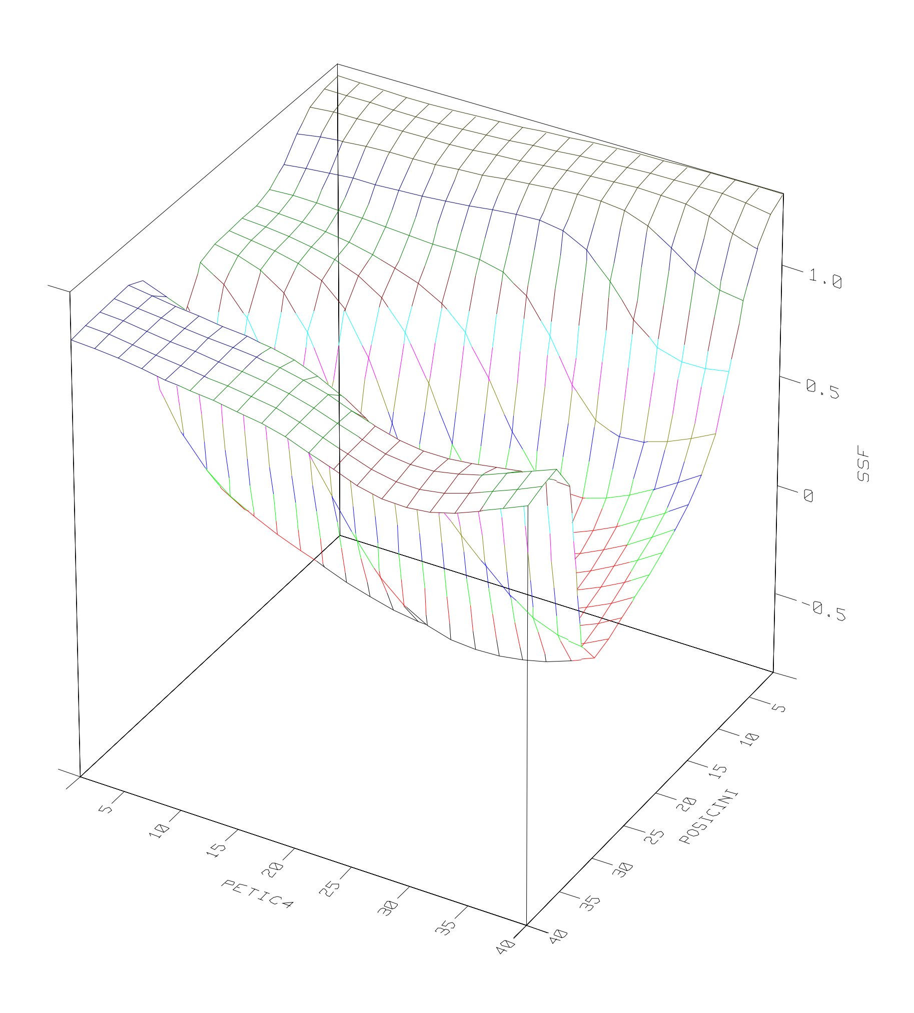


Figura 19.7: Ejemplo de pantalla Nnmodel.

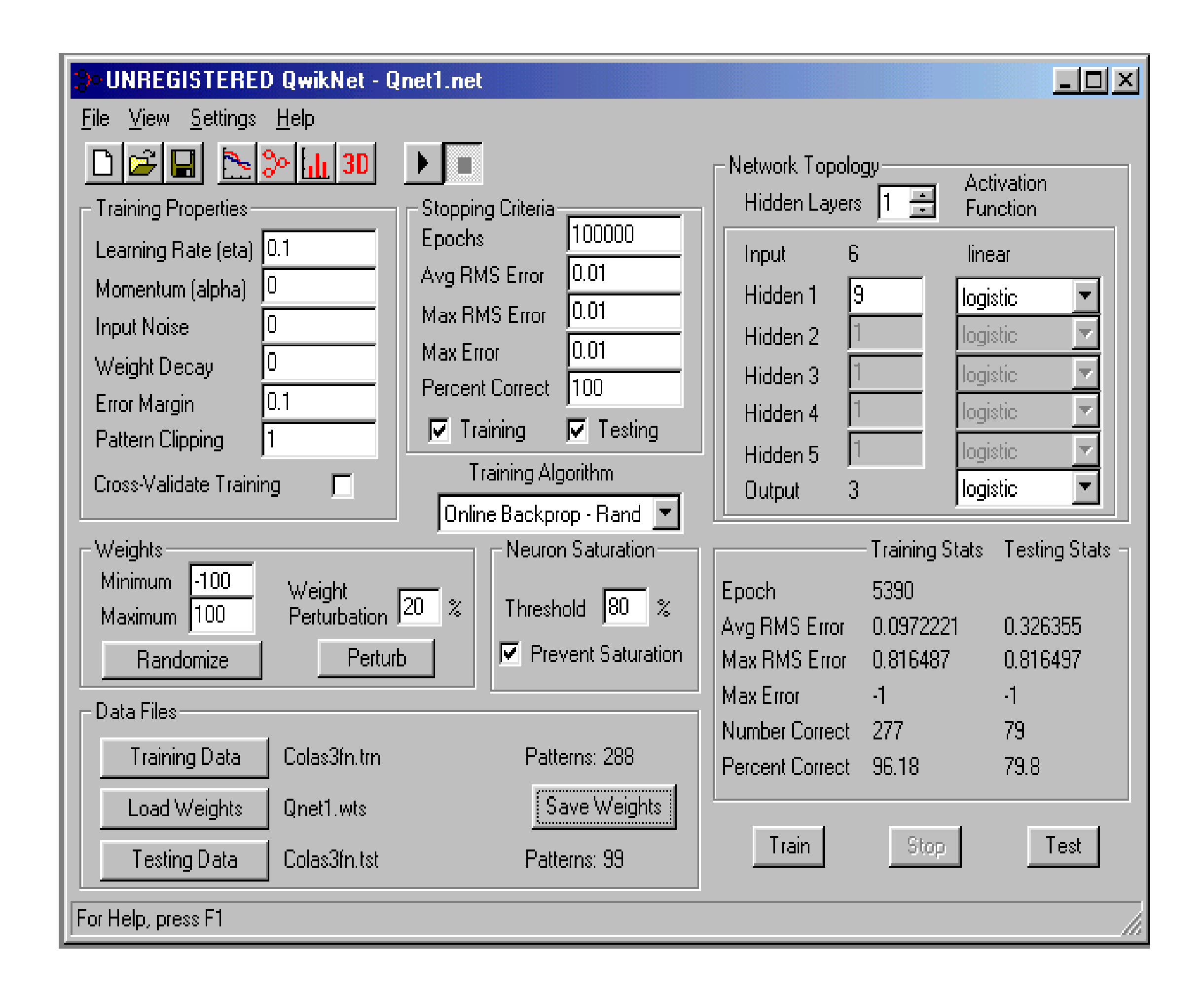


Figura 19.8: Ejemplo de pantalla Qnet.

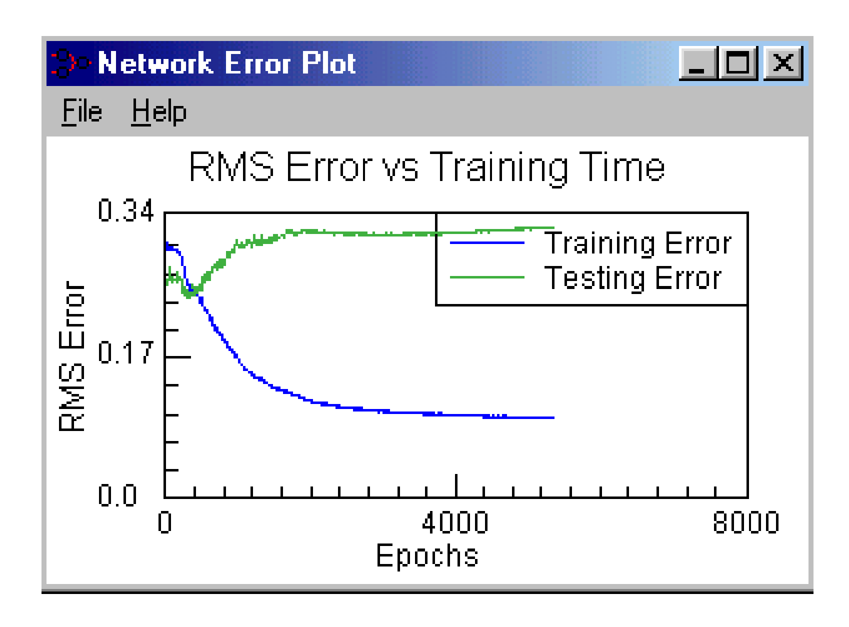


Figura 19.9: Ejemplo de pantalla Qnet.

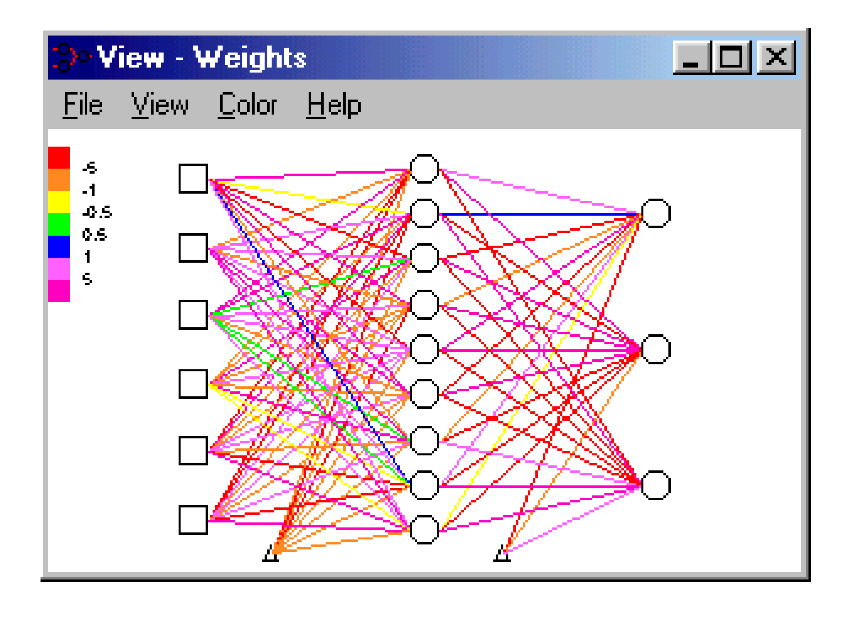


Figura 19.10: Ejemplo de pantalla Qnet.

Estudio y evaluación con la herramienta Nndt

Prueba N°	Configuración de Red	N° de Neuronas Ocultas	Archivo del Modelo	Archivo de Log	Archivo de Salida
1	6:9:6:3	15	Nndt1.mlp	Log1.dat	Salida1.dat
2	6:6:3:3	9	Nndt2.mlp	Log2.dat	Salida2.dat
3	6:4:2:3	6	Nndt3.mlp	Log3.dat	Salida3.dat
4	06:00:03	0	Nndt4.mlp	Log4.dat	Salida4.dat
5	06:05:03	5	Nndt5.mlp	Log5.dat	Salida5.dat
6	6:5:4:3	9	Nndt6.mlp	Log6.dat	Salida6.dat
7	06:04:03	4	Nndt7.mlp	Log7.dat	Salida7.dat

Tabla 19.2: Utilización de la herramienta Nndt.

Prueba N°	Datos de Entrenamiento	Datos de Test	N° de Reg. de Entrenamiento	N° de Reg. de Test	N° de Iteraciones
1	Colas3fn.dat	Colas3ft.dat	177	210	345
2	Colas3fn.dat	Colas3ft.dat	177	210	274
3	Colas3fn.dat	Colas3ft.dat	177	210	521
4	Colas3fn.dat	Colas3ft.dat	177	210	4
5	Colas3fn.dat	Colas3ft.dat	177	210	31
6	Colas3fn.dat	Colas3ft.dat	177	210	338
7	Colas3fn.dat	Colas3ft.dat	177	210	214
Promedios			177	210	247

Tabla 19.3: Utilización de la herramienta Nndt (continuación).

Los resultados obtenidos se resumen en la Tabla 19.2 de la página 523, en la Tabla 19.3 de la página 523 y en la Tabla 19.4 de la página 524, pudiendo extraerse de las mismas las siguientes conclusiones:

- Se efectuaron pruebas con distinto número de capas ocultas y distinto número de neuronas por capa.
- En todos los casos los archivos de datos de entrenamiento y de testeo fueron los mismos.
- El menor error (0,15679432) se logró con la configuración de red 6:6:3:3, es decir con 2 capas ocultas de 6 y 3 neuronas respectivamente y luego de 274 iteraciones.
- Podría decirse que aún el error mínimo obtenido es relativamente alto.

Estudio y evaluación con la herramienta Nnmodel

Los resultados obtenidos se resumen en la Tabla 19.5 de la página 524, en la Tabla 19.6 de la página 525 y en la Tabla 19.7 de la página 525, pudiendo extraerse de las mismas las siguientes conclusiones:

Prueba N°	Error
1	0,229280308
2	0,156794318
3	0,21210684
4	0,284308086
5	0,279096262
6	0,212079128
7	0,255048114
Promedios	0,232673294

Tabla 19.4: Utilización de la herramienta Nndt (continuación).

Prueba N°	Configuración de Red	N° de Neuronas Ocultas	Archivo del Modelo	Reporte Estadístico	Reporte de Sensitividad
1	06:15:03	15	Nnmodel1.bep	Nnrep1.rpt	Nnrepo1.rpt
2	06:09:03	9	Nnmodel2.bep	Nnrep2.rpt	Nnrepo2.rpt
3	06:06:03	6	Nnmodel3.bep	Nnrep3.rpt	Nnrepo3.rpt
4	06:00:03	0	Nnmodel4.bep	Nnrep4.rpt	Nnrepo4.rpt
5	06:05:03	5	Nnmodel5.bep	Nnrep5.rpt	Nnrepo5.rpt
6	06:09:03	9	Nnmodel6.bep	Nnrep6.rpt	Nnrepo6.rpt
7	06:04:03	4	Nnmodel7.bep	Nnrep7.rpt	Nnrepo7.rpt

Tabla 19.5: Utilización de la herramienta Nnmodel.

- Se efectuaron pruebas con distinto número de neuronas ocultas.
- En todos los casos los archivos de datos fueron los mismos, pero utilizándose un número variable de datos para entrenamiento y para testeo, seleccionando estos últimos de distinta manera.
- El menor error (0,015253) se logró con la configuración de red 6:9:3, es decir con 1 capa oculta de 9 neuronas y luego de 5000 iteraciones.
- Podría decirse que el error mínimo obtenido es relativamente aceptable.
- En pruebas complementarias efectuadas variando otros parámetros tales como learning rate y alpha, no se han obtenido variaciones significativas respecto de los errores indicados en el archivo mencionado.

Estudio y evaluación con la herramienta Qnet

Los resultados obtenidos se resumen en la Tabla 19.8 de la página 526, en la Tabla 19.9 de la página 526 y en la Tabla 19.10 de la página 527, pudiendo extraerse de las mismas las siguientes conclusiones:

Prueba N°	Datos de Entrenamiento	Datos de Test	N° de Reg. de Entrenamiento	N° de Reg. de Test	N° de Iteraciones
1	Colas3fn.txt	0	387	0	1000
2	Colas3fn.txt	Random	311	76	1000
3	Colas3fn.txt	Random	301	86	1000
4	Colas3fn.txt	Secuencial	250	137	1000
5	Colas3fn.txt	Secuencial	125	262	1000
6	Colas3fn.txt	Secuencial	125	262	5000
7	Colas3fn.txt	Random	309	78	5000
Promedios			258	129	2143

Tabla 19.6: Utilización de la herramienta Nnmodel (continuación).

Prueba N°	Error
1	0,066045
2	0,06085
3	0,061114
4	0,104912
5	0,057495
6	0,015253
7	0,063339
Promedios	0,061286857

Tabla 19.7: Utilización de la herramienta Nnmodel (continuación).

- Se efectuaron pruebas con distinto número de capas ocultas y distinto número de neuronas por capa.
- En todos los casos los archivos de datos fueron los mismos, pero utilizándose un número variable de datos para entrenamiento y para testeo, seleccionando estos últimos de distinta manera.
- El menor error (0,159792) se logró con la configuración de red 6:5:3, es decir con 1 capa oculta de 5 neuronas y luego de 10000 iteraciones.
- Podría decirse que el error mínimo obtenido es relativamente aceptable.
- En pruebas complementarias efectuadas variando otros parámetros tales como learn rate y momentum, no se han obtenido variaciones significativas respecto de los errores indicados en el archivo mencionado.

Resumen del estudio y evaluación con las herramientas de RNA

La comparación de los casos en que se ha obtenido el menor error para cada herramienta analizada se indica en la Tabla 19.11 de la página 527, pudiendo extraerse del mismo la

Prueba N°	Configuración de Red	N° de Neuronas Ocultas	Archivo del Modelo	Reporte de Errores	Reporte de Salida
1	6:9:6:3	15	Qnet1.net	Qnet1.err	Qnet1.out
2	6:6:3:3	9	Qnet2.net	Qnet2.err	Qnet2.out
3	6:4:2:3	6	Qnet3.net	Qnet3.err	Qnet3.out
4	06:03:03	3	Qnet4.net	Qnet4.err	Qnet4.out
5	06:05:03	5	Qnet5.net	Qnet5.err	Qnet5.out
6	6:5:4:3	9	Qnet6.net	Qnet6.err	Qnet6.out
7	06:04:03	4	Qnet7.net	Qnet7.err	Qnet7.out

Tabla 19.8: Utilización de la herramienta Qnet.

Prueba N°	Datos de Entrenamiento	Datos de Test	N° de Reg. de Entrenamiento	N° de Reg. de Test	N° de Iteraciones
1	Colas3fn.txt	Random	337	50	10000
2	Colas3fn.txt	Random	287	100	5000
3	Colas3fn.txt	Random	237	150	5000
4	Colas3fn.txt	Fin	337	50	10000
5	Colas3fn.txt	Fin	337	50	10000
6	Colas3fn.txt	Random	317	70	10000
7	Colas3fn.txt	Random	317	70	10000
Promedios			310	77	8571

Tabla 19.9: Utilización de la herramienta Qnet (continuación).

conclusión final de que para los datos analizados, las configuraciones de red empleadas, las herramientas consideradas y los parámetros utilizados, el menor error corresponde a la herramienta *Nnmodel.*

Asimismo puede concluirse que la utilización de tecnología de *redes neuronales artificiales* implantada en el firmware de los controladores de operaciones de entrada / salida permitiría de una manera muy rápida y a un muy bajo nivel, seleccionar cuál sería el algoritmo de planificación de la búsqueda en disco más adecuado para cada conjunto de peticiones pendientes en un momento dado, sin la intervención del sistema operativo para la ejecución del algoritmo de planificación por software, teniendo además en cuenta que para cada conjunto de peticiones la RNA podría de hecho aplicar el algoritmo más adecuado, en vez de siempre el mismo algoritmo ejecutado a nivel del sistema operativo, como frecuentemente ocurre, liberando además ciclos de cpu.

Otra posibilidad sería considerar el desarrollo de un sistema experto que tomara las decisiones que acá se dejan en el ámbito de la RNA, con lo que se lograría mayor precisión en los resultados, es decir que sería mayor el número de aciertos en seleccionar el algoritmo óptimo para cada caso, pero posiblemente se perdería en performance debido a la mayor cantidad y complejidad del código que debería ejecutar el sistema experto.

Prueba N°	Error RMS de Entrenamiento	Error RMS de Testeo	Correlación de Entrenamiento	Correlación de Testeo
1	0,129079	0,279679	0,92033	0,639145
2	0,195188	0,20041	0,806298	0,794543
3	0,201248	0,188811	0,792498	0,820302
4	0,178818	0,146925	0,840516	0,896298
5	0,159792	0,191637	0,876386	0,824044
6	0,197801	0,187039	0,800428	0,823899
7	0,168869	0,196328	0,859185	0,806494
Promedios	0,175827857	0,198689857	0,842234429	0,800675

Tabla 19.10: Utilización de la herramienta Qnet (continuación).

Herramienta	Configuración de Red	N° de Iteraciones	Error	Error Promedio de las Pruebas
Nndt	6:6:3:3	274	0,15679432	0,23267329
Nnmodel	06:09:03	5000	0,015253	0,06128686
Qnet	6:9:6:3	10000	0,129079	0,17582786

Tabla 19.11: Resumen comparativo de la utilización de RNA.

Capítulo 20

Concurrencia e Hilos con Java

20.1 Introducción

Para el desarrollo del presente caso de estudio se ha utilizado el lenguaje **Java,** una muy poderosa herramienta para el *desarrollo orientado a objetos*, la utilización a través de *Internet*, el *desarrollo multiplataforma* y especialmente para la explotación de la *concurrencia a nivel de programa*, mediante la utilización de *hilos de ejecución* o *procesos ligeros*, según se los designa en la bibliografía existente [25, Tanenbaum] y [20, Castillo, Cobo, Gómez y Solares].

Los *hilos* o *procesos ligeros* son una parte de código o miniprograma que puede ser ejecutada independientemente, de forma que una aplicación o un applet puede tener varios hilos ejecutándose simultáneamente y efectuando distintas tareas; estos hilos se encuentran dentro de un programa y son parte de él.

Los *hilos*, a veces también llamados *contextos de ejecución*, pueden ser utilizados para la implementación de algoritmos paralelos o procesos concurrentes, sin ser necesario disponer de equipos con estructura de multiprocesador. En el caso de un solo procesador, los procesos ligeros incorporan mecanismos para compartirlo, estableciéndose prioridades entre ellos y también facilidades de sincronización, cuando es necesario.

Los aspectos teóricos relacionados con los *hilos (threads)* han sido desarrollados en el Capítulo *"Procesos y Procesadores en Sistemas Distribuidos".*

20.2 Objetivo del Caso de Estudio

Conforme a lo antes indicado, el objetivo del caso de estudio desarrollado fue el de *programar una aplicación con Java* (**Hilos.java**) que *implementara el problema de "procesos productores y consumidores"* y que permitiera *generar un archivo reutilizable de seguimiento* (**Hilos.txt** y su versión formateada **Hilos.dat**) y posteriormente efectuar análisis de la forma en que se ha desarrollado la ejecución concurrente de los hilos y la sincronización de los mismos, que es el verdadero objetivo, más allá de la utilización como ejemplo del problema de productores y consumidores.

20.3 Descripción del Problema Planteado

Una descripción detallada del problema de los *hilos de ejecución* puede consultarse en el Capítulo *"Procesos y Procesadores en Sistemas Distribuidos"*, siendo especialmente pertinentes los temas relacionados con *Introducción a los Hilos, Uso de Hilos, Aspectos del Diseño de un Paquete de Hilos* e *Implantación de un Paquete de Hilos.*

20.4 Descripción de los Algoritmos Utilizados

Al igual que en el apartado anterior, una descripción detallada de los fundamentos teóricos de los algoritmos aplicables a este caso, puede consultarse en el Capítulo *"Procesos y Procesadores en Sistemas Distribuidos"*, siendo especialmente pertinentes los temas sobre *Aspectos del Diseño de un Paquete de Hilos* e *Implantación de un Paquete de Hilos.* [25, Tanenbaum].

Respecto del *algoritmo de productores y consumidores* utilizado como base para el desarrollo del programa del caso de estudio, puede ser consultado en los ejemplos sobre el tema de [20, Castillo, Cobo, Gómez y Solares] y además en [23, Tanenbaum].

Los *principales aspectos del algoritmo* desarrollado son los siguientes:

- Los *procesos productores* simulan generar, es decir "grabar" información en un grupo de buffers disponibles.
- Los *procesos consumidores* simulan retirar, es decir "leer" información del grupo de buffers disponibles, que previamente debió ser "cargado" por los procesos productores.
- Los procesos consumidores solo pueden "leer" información previamente "grabada" por los procesos productores.
- Si cuando un proceso consumidor intenta leer no hay suficientes buffers disponibles, debe esperar a que alguno de los procesos productores grabe los buffers.
- Los procesos productores solo actúan cuando el nivel de buffers grabados está por debajo de cierto límite, llamado límite de reposición, que es un dato variable que se ingresa como parámetro de configuración de la simulación, al igual que la cantidad de buffers grabados en cada reposición.
- Los procesos consumidores pueden retirar (leer) buffers en número variable pero hasta cierto límite, que también es un dato variable que se ingresa como parámetro de configuración de la simulación.
- Debido a que los procesos productores solo actúan cuando el número de buffers disponibles está por debajo del nivel de reposición y a que la cantidad de reposición es finita, se asegura que el número de buffers del pool será siempre finito.
- El tiempo que dura la simulación también es un parámetro de configuración que se ingresa como dato expresado en milisegundos.

20.5 Programa Desarrollado

El programa en Java desarrollado se encuentra en el archivo **Hilos.java** y posee las siguientes características:

Se lo puede invocar mediante *"java Hilos"* y recibe como entradas por pantalla los *datos de configuración* de la simulación referidos a:

- Número inicial de buffers ocupados.
- Número máximo de buffers leidos por vez.
- Número mínimo de buffers ocupados antes de grabar más.
- Número de buffers grabados por vez.
- Número de milisegundos de la ejecución.

Respecto del *archivo generado*, el mismo tendrá un tamaño variable según los datos que se hayan suministrado a la ejecución que lo produce, especialmente el tiempo de ejecución; es preciso señalar además que el archivo es reutilizable, es decir que se borra automáticamente al iniciar cada ejecución.

El *código del programa desarrollado* (**Hilos.java**) es el siguiente:

```
// Trabajo práctico parte de la Tesis de la Maestría en Informática y
// Computación - 2000. David Luis la Red Martínez.
//
// Ejemplo de concurrencia e hilos con JAVA.
//
// El ejemplo contempla el problema de productores / consumidores:
// * Los hilos productores graban (y ocupan) buffers en un pool de buffers disponibles.
// * Los hilos consumidores leen (y liberan) buffers del pool de buffers disponibles.
//
import java.io.*;
import java.util.*;
import java.util.Date;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
```

```
import java.applet.*;
import java.applet.Applet;
import java.math.*;
import java.text.*;
import java.lang.String;
import java.lang.*;
public class Hilos extends Frame
implements ActionListener {
VentanaDialogCH ventana;
SimpleDialogCH dialog;
TextArea textArea;
Button boton1;
public Hilos() {
MenuBar barra;
Menu cargadatos, ayuda, acerca, salir;
MenuItem cardatos, ejecuc, listaeje, ayulis, acercade, fin;
// Menú principal.
barra = new MenuBar();
setMenuBar(barra);
// Agregado de submenú al principal.
cargadatos = new Menu("Configuración y Ejecución", true);
barra.add(cargadatos);
// Creación de opciones del menú.
cardatos = new MenuItem("Datos de Configuración");
cardatos.addActionListener(this);
cargadatos.add(cardatos);
ejecuc = new MenuItem("Ejecución Concurrente e Hilos");
ejecuc.addActionListener(this);
```

```
cargadatos.add(ejecuc);
listaeje = new MenuItem("Lista Resultados de Ejecución");
listaeje.addActionListener(this);
cargadatos.add(listaeje);
// Agregado del submenú al principal.
ayuda = new Menu("Ayuda", true);
barra.add(ayuda);
// Creación de opciones del menú.
ayulis = new MenuItem("Listar Ayuda");
ayulis.addActionListener(this);
ayuda.add(ayulis);
// Agregado del submenú al principal.
salir = new Menu("Salir", true);
barra.add(salir);
// Creación de opciones del menú.
fin = new MenuItem("Salir del Sistema");
fin.addActionListener(this);
salir.add(fin);
// Agregado del submenú al principal.
acerca = new Menu("Acerca de", true);
barra.add(acerca);
// Creación de opciones del menú.
acercade = new MenuItem("Acerca del Sistema");
acercade.addActionListener(this);
acerca.add(acercade);
// Habilitación del cierre de la ventana.
addWindowListener(new WindowAdapter() {
public void windowClosing(WindowEvent e) {
```

```
System.exit(0);
}
});
}
public void actionPerformed(ActionEvent evt) {
MenuItem c = (MenuItem) evt.getSource();
String arg = c.getLabel();
if(arg.equals(”Salir del Sistema”)) {
System.exit(0);
}
else if(arg.equals(”Datos de Configuración”)) {
// Para Datos de Configuración.
// Lee datos de entrada.
// Crea etiqueta para documento.
VentanaDialogCH ventana=new VentanaDialogCH();
ventana.setTitle(”Carga de Datos de Configuración”);
ventana.setVisible(true);
ventana.setBackground(Color.yellow);
ventana.setForeground(Color.red);
ventana.setSize(new Dimension(600,400));
}
else if(arg.equals(”Ejecución Concurrente e Hilos”)) {
// Para Ejecución Concurrente e Hilos.
// Lee datos de entrada.
// Crea etiqueta para documento.
VentanaDialogECH ventana=new VentanaDialogECH();
ventana.setTitle(”Realiza Ejecución Concurrente e Hilos”);
ventana.setVisible(true);
```

```
ventana.setBackground(Color.red);
ventana.setForeground(Color.yellow);
ventana.setSize(new Dimension(600,400));
}
else if(arg.equals("Lista Resultados de Ejecución")) {
// Para Lista Resultados de Ejecución.
// Lee datos de entrada.
// Crea etiqueta para documento.
VentanaDialogLCH ventana=new VentanaDialogLCH();
ventana.setTitle("Lista Resultados de Ejecución Concurrente e Hilos");
ventana.setVisible(true);
ventana.setBackground(Color.green);
ventana.setForeground(Color.orange);
ventana.setSize(new Dimension(600,400));
}
else if(arg.equals("Listar Ayuda")) {
// Para Listar Ayuda.
// Lee datos de entrada.
// Crea etiqueta para Listar Ayuda.
VentanaDialogA ventana=new VentanaDialogA();
ventana.setTitle("Lista Ayuda");
ventana.setVisible(true);
ventana.setBackground(Color.cyan);
ventana.setForeground(Color.magenta);
ventana.setSize(new Dimension(600,400));
}
else if(arg.equals("Acerca del Sistema")) {
// Para Acerca del Sistema.
```

```
// Lee datos de entrada.
// Crea etiqueta para Acerca del Sistema.
VentanaDialogAS ventana=new VentanaDialogAS();
ventana.setTitle("Acerca del Sistema");
ventana.setVisible(true);
ventana.setBackground(Color.orange);
ventana.setForeground(Color.blue);
ventana.setSize(new Dimension(600,400));
}
}
public static void main (String args[]) {
// Creación del menú principal.
Hilos menus = new Hilos();
menus.setTitle("Ejecución Concurrente e Hilos con JAVA");
menus.setVisible(true);
menus.setBackground(Color.cyan);
menus.setSize(new Dimension(850,650));
// Creación de la imágen dinámica.
VImaCanvas imagen1 = new VImaCanvas();
imagen1.setTitle("???????");
imagen1.setVisible(true);
imagen1.setBackground(Color.orange);
imagen1.setSize(new Dimension(100,100));
}
}
class VentanaDialogCH extends Frame implements ActionListener{
SimpleDialogCH dialog;
TextArea textArea;
```

```
Button boton1;
public VentanaDialogCH(){
textArea=new TextArea(5,40);
textArea.setEditable(false);
add("Center",textArea);
boton1=new Button("Diálogo p/ configuración");
boton1.addActionListener(this);
Panel panel=new Panel();
panel.add(boton1);
add("South",panel);
panel.setBackground(Color.yellow);
addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e){
setVisible(false);
}});
}
public void actionPerformed(ActionEvent evt){
String str=evt.getActionCommand();
if(str.equals("Diálogo p/ configuración")){
if(dialog==null)
dialog=new SimpleDialogCH(this,"Configuración");
dialog.show();
}
}
public void setText1(String stock){
try {
textArea.setFont(new Font("Times", 1, 11));
textArea.setForeground(Color.orange);
```

```
String asterisc = ”\n************************************”;
textArea.append(asterisc);
textArea.append(”\nNúmero inicial de buffers ocupados: ” + stock + ”.”);
}
catch (java.lang.Exception ex) {
// Atrapa excepciones y las despliega.
ex.printStackTrace ();
}
try {
File f0 = new File (”Hilos.txt”);
f0.delete();
RandomAccessFile f1 = new RandomAccessFile (”Hilos.txt”, ”rw”);
long longitud;
longitud = 0;
f1.seek(longitud);
f1.writeUTF(stock);
f1.close();
}
catch (Exception e) {
System.out.println(”Archivo Hilos: Salida en grabación de parámetros: ” + e);
}
}
public void setText2(String maxcompra){
try {
textArea.append(”\nNúmero máximo de buffers leidos por vez: ” + maxcompra + ”.”);
}
catch (java.lang.Exception ex) {
// Atrapa excepciones y las despliega.
```

```
ex.printStackTrace ();
}
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
longitud = f1.length();
f1.seek(longitud);
f1.writeUTF(maxcompra);
f1.close();
}
catch (Exception e) {
System.out.println("Archivo Hilos: Salida en grabación de parámetros: " + e);
}
}
public void setText3(String limitereponer){
try {
textArea.append("\nNúmero mínimo de buffers ocupados antes de grabar más: " +
    limitereponer + ".");
}
catch (java.lang.Exception ex) {
// Atrapa excepciones y las despliega.
ex.printStackTrace ();
}
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
longitud = f1.length();
f1.seek(longitud);
f1.writeUTF(limitereponer);
```

```
f1.close();
}
catch (Exception e) {
System.out.println(”Archivo Hilos: Salida en grabación de parámetros: ” + e);
}
}
public void setText4(String cantidadreponer){
try {
textArea.append(”\nNúmero de buffers grabados por vez: ” + cantidadreponer + ”.”);
}
catch (java.lang.Exception ex) {
// Atrapa excepciones y las despliega.
ex.printStackTrace ();
}
try {
RandomAccessFile f1 = new RandomAccessFile (”Hilos.txt”, ”rw”);
long longitud;
longitud = f1.length();
f1.seek(longitud);
f1.writeUTF(cantidadreponer);
f1.close();
}
catch (Exception e) {
System.out.println(”Archivo Hilos: Salida en grabación de parámetros: ” + e);
}
}
public void setText5(String tiempoapertura){
try {
```

```
String asterisc = "\n************************************";
textArea.append("\nNúmero de milisegundos de la simulación: " + tiempoapertura +
    ".");
textArea.append(asterisc);
textArea.append("\n ");
}
catch (java.lang.Exception ex) {
// Atrapa excepciones y las despliega.
ex.printStackTrace ();
}
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
longitud = f1.length();
f1.seek(longitud);
f1.writeUTF(tiempoapertura);
f1.close();
}
catch (Exception e) {
System.out.println("Archivo Hilos: Salida en grabación de parámetros: " + e);
}
}
}
class SimpleDialogCH extends Dialog implements ActionListener{
TextField tstock, tmaxcompra, tlimitereponer, tcantidadreponer, ttiempoapertura;
VentanaDialogCH parent;
Button b1;
public String stock;
public String MAXCOMPRA;
```

```
public String LIMITEREPONER;
public String CANTIDADREPONER;
public String TIEMPOAPERTURA;
public SimpleDialogCH(Frame fr,String titulo){
super(fr,titulo,true);
parent=(VentanaDialogCH)fr;
Panel p1=new Panel();
Label etiqueta1=new Label("Número inicial de buffers ocupados (0):");
p1.add(etiqueta1);
tstock=new TextField("0",4);
p1.add(tstock);
add("Center",p1);
Label etiqueta2=new Label("Número máximo de buffers leidos por vez (30):");
p1.add(etiqueta2);
tmaxcompra=new TextField("30",4);
p1.add(tmaxcompra);
add("Center",p1);
Label etiqueta3=new Label("Número mínimo de buffers ocupados antes de grabar más
    (40):");
p1.add(etiqueta3);
tlimitereponer=new TextField("40",4);
p1.add(tlimitereponer);
add("Center",p1);
Label etiqueta4=new Label("Número de buffers grabados por vez (50):");
p1.add(etiqueta4);
tcantidadreponer=new TextField("50",4);
p1.add(tcantidadreponer);
add("Center",p1);
Label etiqueta5=new Label("Número de milisegundos de la ejecución (2000):");
```

```
p1.add(etiqueta5);
ttiempoapertura=new TextField(”2000”,4);
p1.add(ttiempoapertura);
add(”Center”,p1);
Panel p2=new Panel();
p2.setLayout(new FlowLayout(FlowLayout.RIGHT));
b1=new Button(”Cancelar”);
b1.addActionListener(this);
Button b2=new Button(”Aceptar”);
b2.addActionListener(this);
p2.add(b1);
p2.add(b2);
add(”South”,p2);
setSize(500,200);
}
public void actionPerformed(ActionEvent evt){
String str=evt.getActionCommand();
if(str.equals(”Aceptar”))
parent.setText1(tstock.getText());
stock = tstock.getText();
parent.setText2(tmaxcompra.getText());
MAXCOMPRA = tmaxcompra.getText();
parent.setText3(tlimitereponer.getText());
LIMITEREPONER = tlimitereponer.getText();
parent.setText4(tcantidadreponer.getText());
CANTIDADREPONER = tcantidadreponer.getText();
parent.setText5(ttiempoapertura.getText());
TIEMPOAPERTURA = ttiempoapertura.getText();
```

```
setVisible(false);
}
}
class VentanaDialogECH extends Frame implements ActionListener{
TextArea textArea;
public VentanaDialogECH(){
textArea=new TextArea(5,40);
textArea.setEditable(false);
add("Center",textArea);
String stock, maxcompra, limitereponer, cantidadreponer, tiempoapertura;
stock = ""; maxcompra = ""; limitereponer = "";
cantidadreponer = ""; tiempoapertura = "";
addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e){
setVisible(false);
}});
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
longitud = 0;
f1.seek(longitud);
stock = f1.readUTF();
maxcompra = f1.readUTF();
limitereponer = f1.readUTF();
cantidadreponer = f1.readUTF();
tiempoapertura = f1.readUTF();
f1.close();
}
```

```
catch (Exception e) {
System.out.println("Archivo Hilos: Salida luego de leer parámetros para actualizar datos:
" + e);
}
try {
Almacen a = new Almacen();
Productor p1 = new Productor(a,1);
Productor p2 = new Productor(a,2);
Consumidor c1 = new Consumidor(a,1);
Consumidor c2 = new Consumidor(a,2);
Consumidor c3 = new Consumidor(a,3);
a.stock = Integer.parseInt(stock);
a.MAXCOMPRA = Integer.parseInt(maxcompra);
a.LIMITEREPONER = Integer.parseInt(limitereponer);
a.CANTIDADREPONER = Integer.parseInt(cantidadreponer);
a.TIEMPOAPERTURA = Integer.parseInt(tiempoapertura);
p1.start();
p2.start();
c1.start();
c2.start();
c3.start();
}
catch (java.lang.Exception ex) {
// Atrapa excepciones y las despliega.
ex.printStackTrace ();
}
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
```

```
String c, d, asterisc;
longitud = 0;
f1.seek(longitud);
asterisc = ”\n************************************”;
d = ”\n ”;
textArea.setFont(new Font(”Times”, 1, 11));
textArea.setForeground(Color.red);
textArea.append(asterisc);
textArea.append(”\nInicio de la ejecución concurrente con hilos.”);
textArea.append(”\nLos datos de configuración son los siguientes: ”);
c = f1.readUTF();
textArea.append(”\nNúmero inicial de buffers ocupados: ” + c + ”.”);
c = f1.readUTF();
textArea.append(”\nNúmero máximo de buffers leidos por vez: ” + c + ”.”);
c = f1.readUTF();
textArea.append(”\nNúmero mínimo de buffers ocupados antes de grabar más: ” + c +
    ”.”);
c = f1.readUTF();
textArea.append(”\nNúmero de buffers grabados por vez: ” + c + ”.”);
c = f1.readUTF();
textArea.append(”\nNúmero de milisegundos de la ejecución: ” + c + ”.”);
textArea.append(asterisc);
textArea.append(d);
f1.close();
}
catch (Exception e) {
System.out.println(”Archivo Hilos: Salida luego de leer parámetros para desplegarlos: ”
    + e);
}
```

```
}
public void actionPerformed(ActionEvent evt){
String str=evt.getActionCommand();
}
}
class VentanaDialogLCH extends Frame implements ActionListener{
TextArea textArea;
public VentanaDialogLCH(){
textArea=new TextArea(5,40);
textArea.setEditable(false);
add("Center",textArea);
addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e){
setVisible(false);
}});
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
String c, d, asterisc;
longitud = 0;
f1.seek(longitud);
asterisc = "\n**************************************";
d = "\n ";
textArea.setFont(new Font("Times", 1, 11));
textArea.setForeground(Color.red);
textArea.append(asterisc);
textArea.append("\nInicio de la ejecución concurrente con hilos.");
textArea.append("\nLos datos de configuración son los siguientes: ");
```

```
c = f1.readUTF();
textArea.append("\nNúmero inicial de buffers ocupados: " + c + ".");
c = f1.readUTF();
textArea.append("\nNúmero máximo de buffers leidos por vez: " + c + ".");
c = f1.readUTF();
textArea.append("\nNúmero mínimo de buffers ocupados antes de grabar más: " + c +
    ".");
c = f1.readUTF();
textArea.append("\nNúmero de buffers grabados por vez: " + c + ".");
c = f1.readUTF();
textArea.append("\nNúmero de milisegundos de la ejecución: " + c + ".");
textArea.append(asterisc);
textArea.append(d);
while ((c = f1.readUTF()) != null) {
textArea.append(c);
textArea.append(d);
}
f1.close();
}
catch (Exception e) {
System.out.println("Archivo Hilos: Salida luego de leer parámetros para desplegarlos: "
    + e);
}
}
public void actionPerformed(ActionEvent evt){
String str=evt.getActionCommand();
}
}
class VentanaDialogA extends Frame implements ActionListener{
```

```
TextArea textArea;
public VentanaDialogA(){
textArea=new TextArea(5,40);
textArea.setEditable(false);
add("Center",textArea);
addWindowListener(new WindowAdapter(){
public void windowClosing(WindowEvent e){
setVisible(false);
}});
try {
String asterisc = "\n ********************************************";
textArea.setFont(new Font("Times", 1, 11));
textArea.setForeground(Color.blue);
textArea.append("\n ");
textArea.append(asterisc);
textArea.append("\n ");
textArea.append("\n Ayuda para la ejecución concurrente e hilos con JAVA.");
textArea.append("\n *************************************************");
textArea.append("\n ");
textArea.append("\n El ejemplo contempla el problema de productores / consumido-
    res:");
textArea.append("\n * Los hilos productores graban (y ocupan) buffers en un pool");
textArea.append("\n de buffers disponibles.");
textArea.append("\n * Los hilos consumidores leen (y liberan) buffers del pool");
textArea.append("\n de buffers disponibles.");
textArea.append("\n ");
textArea.append("\n Las opciones previstas contemplan lo siguiente:");
textArea.append("\n * Carga de datos de configuración de la ejecución.");
textArea.append("\n * Ejecución propiamente dicha.");
```

```
textArea.append("\n * Listado de los resultados por pantalla.");
textArea.append("\n ");
textArea.append("\n Los datos se cargan interactivamente por pantalla, ");
textArea.append("\n donde se sugieren valores de referencia.");
textArea.append("\n ");
textArea.append("\n Los resultados se graban en el archivo Hilos.txt,");
textArea.append("\n como registros de longitud variable.");
textArea.append("\n ");
textArea.append("\n ");
textArea.append(asterisc);
textArea.append("\n ");
}
catch (java.lang.Exception ex) {
// Atrapa otro tipo de excepciones y las despliega.
ex.printStackTrace ();
}
}
public void actionPerformed(ActionEvent evt){
String str=evt.getActionCommand();
}
}
class VentanaDialogAS extends Frame implements ActionListener{
TextArea textArea;
public VentanaDialogAS(){
textArea=new TextArea(5,40);
textArea.setEditable(false);
add("Center",textArea);
addWindowListener(new WindowAdapter(){
```

```
public void windowClosing(WindowEvent e){
setVisible(false);
}});
try {
String asterisc = "\n ***********************************************";
textArea.setFont(new Font("Times", 1, 11));
textArea.setForeground(Color.black);
textArea.append("\n ");
textArea.append(asterisc);
textArea.append("\n ");
textArea.append("\n Acerca del sistema de concurrencia e hilos con JAVA.");
textArea.append("\n **************************************************");
textArea.append("\n ");
textArea.append("\n Trabajo práctico de Sistemas Operativos.");
textArea.append("\n Licenciatura en Sistemas.");
textArea.append("\n Universidad Nacional del Nordeste (Argentina).");
textArea.append("\n ");
textArea.append("\n ");
textArea.append(asterisc);
textArea.append("\n ");
}
catch (java.lang.Exception ex) {
// Atrapa otro tipo de excepciones y las despliega.
ex.printStackTrace ();
}
}
public void actionPerformed(ActionEvent evt){
String str=evt.getActionCommand();
```

```
}
}
class VImaCanvas extends Frame
implements AdjustmentListener {
Image imagen1;
ImaCanvas imaani;
Panel panel;
public VImaCanvas(){
setLayout(new BorderLayout());
panel = new Panel();
// Agregado de la imagen animada.
imagen1 = Toolkit.getDefaultToolkit().getImage("Ani0076.gif");
imaani = new ImaCanvas(imagen1);
panel.add(imaani);
add("Center",imaani);
}
public void adjustmentValueChanged(AdjustmentEvent evt){
imaani.repaint();
}
}
class ImaCanvas extends Canvas{
Image img;
int x,y;
public ImaCanvas(Image imagen1){
img=imagen1;
x=0;
y=0;
}
```

```
public void paint(Graphics g){
g.drawImage(img,0,0,this);
}
}
class Productor extends Thread {
private Almacen almacen;
private int numero;
public Productor(Almacen a, int numero) {
almacen = a;
this.numero = numero;
}
public void run() {
while(!almacen.cerrado){
if(System.currentTimeMillis()
-almacen.hora>almacen.TIEMPOAPERTURA){
almacen.cerrado=true;
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
String c;
c = "Fin de la ejecución.";
longitud = f1.length();
f1.seek(longitud);
f1.writeUTF(c);
f1.close();
}
catch (Exception e) {
System.out.println("Archivo Hilos: Salida luego de grabar Fin de la ejecución: " + e);
```

```
}
}
almacen.reponer(numero);
try {
sleep((int)(Math.random() * 100));
}
catch (InterruptedException e) {
}
}
}
}
class Consumidor extends Thread {
private Almacen almacen;
private int numero;
public Consumidor(Almacen a, int numero) {
almacen = a;
this.numero = numero;
}
public void run() {
int value;
while(!almacen.cerrado){
value=(int)(Math.random()*almacen.MAXCOMPRA);
almacen.comprar(numero,value);
}
}
}
class Almacen {
public int stock=0;
```

```
private boolean necesitareponer;
public boolean cerrado=false;
public int MAXCOMPRA=0;
public int LIMITEREPONER=0;
public int CANTIDADREPONER=0;
public long TIEMPOAPERTURA=0;
long hora=System.currentTimeMillis();
public synchronized void comprar(int consumidor,
int cantidad) {
while (necesitareponer == true) {
try { wait();
} catch (InterruptedException e) {
}
}
if(!cerrado){
if(stock<cantidad){
cantidad=stock;
}
stock-=cantidad;
try {
RandomAccessFile f1 = new RandomAccessFile ("Hilos.txt", "rw");
long longitud;
String c;
c = "Quedan " + stock + " unidades. El Consumidor " + consumidor + " ha leido "
    +cantidad+" unidades.";
longitud = f1.length();
f1.seek(longitud);
f1.writeUTF(c);
f1.close();
```

```
}
catch (Exception e) {
System.out.println(”Archivo Hilos: Salida luego de grabar Hilo Consumidor: ” + e);
}
if(stock<=LIMITEREPONER)
necesitareponer = true;
}
notify();
}
public synchronized void reponer(int productor) {
if (!cerrado){
while (necesitareponer == false) {
try { wait();
} catch (InterruptedException e) {}
}
stock+=CANTIDADREPONER;
try {
RandomAccessFile f1 = new RandomAccessFile (”Hilos.txt”, ”rw”);
long longitud;
String c;
c = ”Quedan ” + stock + ” unidades. El Productor ” + productor + ” ha grabado ” +
    CANTIDADREPONER + ” unidades.”;
longitud = f1.length();
f1.seek(longitud);
f1.writeUTF(c);
f1.close();
}
catch (Exception e) {
System.out.println(”Archivo Hilos: Salida luego de grabar Hilo Productor: ” + e);
```

```
}
necesitareponer = false;
}
notify();
}
}
```

20.6 Datos y Ejecuciones

Los resultados detallados de las ejecuciones se muestran paso a paso en pantalla y son grabados en el archivo de datos y ejecuciones, que en este caso ha sido el archivo **Hilos.txt,** que es un archivo con registros de longitud variable, cuyos datos formateados, para una ejecución tomada como referencia, se encuentran en el archivo **Hilos.dat**.

El archivo generado (**Hilos.txt**), posee la siguiente estructura:

- Número inicial de buffers ocupados.
- Número máximo de buffers leidos por vez.
- Número mínimo de buffers ocupados antes de grabar más.
- Número de buffers grabados por vez.
- Número de milisegundos de la ejecución.
- Un registro por cada intervención de cada proceso productor y de cada proceso consumidor, donde se indica el número actual de buffers disponibles luego de la intervención del proceso correspondiente y el número de buffers grabados o leidos, según corresponda.
- Un registro final de fin de la ejecución.

El contenido del mencionado archivo **Hilos.dat** luego de una de las ejecuciones de evaluación puede consultarse en los Anexos; un detalle parcial es el siguiente:

```
0
30
40
50
2000
Quedan 0 unidades. El Consumidor 1 ha leido 0 unidades.
```

Quedan 50 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 49 unidades. El Consumidor 2 ha leido 1 unidades.

Quedan 31 unidades. El Consumidor 2 ha leido 18 unidades.

Quedan 81 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 53 unidades. El Consumidor 1 ha leido 28 unidades.

Quedan 42 unidades. El Consumidor 1 ha leido 11 unidades.

Quedan 39 unidades. El Consumidor 1 ha leido 3 unidades.

Quedan 89 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 85 unidades. El Consumidor 2 ha leido 4 unidades.

Quedan 66 unidades. El Consumidor 2 ha leido 19 unidades.

Quedan 66 unidades. El Consumidor 2 ha leido 0 unidades.

Quedan 61 unidades. El Consumidor 2 ha leido 5 unidades.

Quedan 33 unidades. El Consumidor 2 ha leido 28 unidades.

Quedan 83 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 58 unidades. El Consumidor 3 ha leido 25 unidades.

Quedan 46 unidades. El Consumidor 3 ha leido 12 unidades.

Quedan 22 unidades. El Consumidor 3 ha leido 24 unidades.

Quedan 72 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 50 unidades. El Consumidor 1 ha leido 22 unidades.

20.7 Resultados y Conclusiones

La utilización de *Java* para la resolución del problema planteado ha resultado muy satisfactoria, destacándose las facilidades y potencia del producto, en especial para el manejo de hilos de ejecución y de la sincronización de los mismos.

Los *resultados obtenidos ratifican*, como era de esperarse, las *previsiones teóricas* en cuanto a las *facilidades* de los *hilos* o *procesos ligeros* para resolver problemas de *concurrencia* y de *simultaneidad*, en caso de disponer de multiprocesadores; esto los hace especialmente aplicables a un gran número de problemas propios de los *sistemas operativos* como administradores de *recursos compartidos*, siendo el caso del problema planteado, es decir el caso de *procesos productores* y *procesos consumidores* solo un caso genérico de la problemática antes mencionada.

Capítulo 21

Anomalía de Belady con Matlab

21.1 Introducción

Para el desarrollo del presente caso de estudio se ha utilizado el lenguaje **Matlab,** una muy poderosa herramienta para el *cálculo numérico*, aplicada acá a la simulación de requerimientos de paginación de memoria virtual y a evaluar si secuencias aleatorias de dichos requerimientos podrían llevar a situaciones del tipo de la llamada *Anomalía de Belady* o *Anomalía FIFO* [7, Deitel].

Lo interesante del presente caso de estudio es que se permiten simular requerimientos de paginación para secuencias grandes (de cientos o de miles de requerimientos) como así también para grupos muy numerosos de celdas de páginas, lo que nos acerca a situaciones reales.

Los aspectos teóricos relacionados con los algoritmos de paginación y con la anomalía mencionada, han sido desarrollados en el Capítulo *"Administración de la Memoria".*

21.2 Objetivo del Caso de Estudio

Conforme a lo antes indicado, el objetivo del caso de estudio desarrollado fue el de *codificar un programa en Matlab* (**Anomalia.m**) que *efectuara el testeo de la posible Anomalía de Belady para una secuencia aleatoria de requerimientos de páginas en un sistema de paginación FIFO,* visualizando en pantalla los datos obtenidos y gráficamente los principales resultados y que permitiera *generar un archivo acumulativo de seguimiento* (**Anomalia.txt** y su versión formateada **Anomalia.dat).**

21.3 Descripción del Problema Planteado

Una descripción detallada del problema de la *Anomalía de Belady* o *Anomalía FIFO* puede consultarse en el Capítulo *"Administración de la Memoria"*, siendo especialmente pertinente el tema relacionado con *Estrategias de Administración del Almacenamiento Virtual.*

21.4 Descripción del Algoritmo Utilizado

Al igual que en el apartado anterior, una descripción detallada de los fundamentos teóricos del algoritmo aplicable a este caso, puede consultarse en el Capítulo *"Administración de la Memoria"*, siendo especialmente pertinente el tema relacionado con *Estrategias de Administración del Almacenamiento Virtual.* [7, Deitel].

21.5 Programa Desarrollado

El programa en Matlab desarrollado se encuentra en el archivo **Anomalia.m** y posee las siguientes características:

Se lo puede invocar desde el entorno de Matlab mediante *"Anomalia.m"* y recibe como entradas por pantalla los *datos de configuración* de la simulación referidos a:

- Número de celdas de página en la memoria real.
- Número de elementos del vector de requerimientos de páginas y número de testeos que se harán en la simulación.

Las salidas por pantalla incluyen:

- Seguimiento del algoritmo de paginación FIFO, con indicación de los fallos de página que se van produciendo.
- Número de fallos de página ocurridos y número de celdas de página de memoria real disponibles para cada ciclo de la simulación, teniendo presente que para cada conjunto de datos de entrada, se efectúan tres simulaciones, una con el 80% del número suministrado como número de celdas de página en la memoria real, otra con el 100% de dicho número y una tercera con el 120% del mismo.
- Conclusión respecto de si se ha producido o no la Anomalía de Belady en cada una de las simulaciones efectuadas.
- Representación gráfica de los principales resultados obtenidos, indicándose el número de fallos de páginas respecto del número de celdas de página, para las distintas simulaciones efectuadas.

Respecto del *archivo generado*, el mismo tendrá un tamaño variable según los datos que se hayan suministrado a las ejecuciones que lo producen; es preciso señalar además que el archivo es acumulativo, es decir que no se borra automáticamente al iniciar cada ejecución.

El *código del programa desarrollado* (**Anomalia.m**) es el siguiente:

```
% Maestría en Informática y Computación.

% Testeo de la posible Anomalía de Belady para una secuencia aleatoria de requerimientos

% de páginas en un sistema de paginación FIFO.
```

```
% n: Número de celdas de página en la memoria real.
% M: Vector de celdas de página en la memoria real.
% R: Matriz de requerimientos de páginas por testeo.
% T: Vector de requerimientos de páginas para un testeo.
% fallos: número de fallos de página.
clc
echo off
n = 0;
M = 0;
R = 0;
T = 0;
fallos = 0;
i = 0;
j = 0;
r = 0;
s = 0;
p = 0;
encontrada = 0;
k = 0;
l = 0;
Fp(3) = 0;
Cp(3) = 0;
Aux(2) = 0;
disp(' ')
disp('Testeo de la posible Anomalía de Belady para una secuencia aleatoria de')
disp('requerimientos de páginas en un sistema de paginación FIFO.')
disp('*************************************************************************')
disp(' ')
```

```
disp('Para indicar el número de celdas de página en la memoria real tipee "n = N°ent;"')
disp('donde "N°ent" es un número entero positivo.')
disp('Luego tipee "return".')
disp(' ')
keyboard
disp('Para generar aleatoriamente la matriz de requerimientos de páginas tipee "R = rand(N°elem,N°test);"')
disp('donde "N°elem" es el número de elementos del vector de requerimientos de páginas y')
disp('"N°test" es el número de testeos que se harán.')
disp('Luego tipee "return".')
disp(' ')
keyboard
for i = 1:n,
M(i) = 0;
end
if R == 0
disp('No se ha cargado la matriz de requerimientos de páginas.')
keyboard
end
if n == 0
disp('No se ha cargado el número de celdas de páginas disponibles.')
keyboard
end
l = n;
r = size(R,1);
s = size(R,2);
for q = 1:s,
for i = 1:r,
```

```
T(i) = round(R(i,q) * 1000);
end
disp('*******************************************************')
disp('T: Vector de requerimientos de páginas para la simulación:')
disp('*******************************************************')
T
% Determinación de si la página requerida está en memoria real.
n = round(l * 0.8);
for k = 1:3,
for j = 1:r,
for i = 1:n,
if M(i) == T(j)
disp('Página encontrada en memoria real: ')
disp(T(j))
encontrada = 1;
i = n;
end
end
if encontrada == 0
disp('Fallo de página; se debe cargar a memoria real la página: ')
disp(T(j))
fallos = fallos + 1;
for i = 1:(n-1),
M(i) = M(i+1);
end
M(n) = T(j);
end
encontrada = 0;
```

```
end
disp('*****************************************************')
disp('Número de fallos de página ocurridos: ')
disp(fallos)
disp('Número de celdas de página de memoria real: ')
disp(n)
disp('*****************************************************')
Fp(k) = fallos;
Cp(k) = n;
fallos = 0;
if k == 1
n = l;
else if k == 2
n = round(l * 1.2);
end
end
for i = 1:n,
M(i) = 0;
end
end
disp(' ')
disp('*****************************************************')
disp('Pares celdas de página disponibles y fallos de páginas ocurridos: ')
rx = 'Vector de requerimientos de páginas para la simulación:';
tx = 'Pares celdas de página disponibles y fallos de páginas ocurridos: ';
bx = ' ';
%
% Colocar C: o D: según el disco de trabajo.
```

```
%
fid = fopen('D:\TesisMic\Anomalia\Anomalia.txt','a');
fprintf(fid,rx);
fprintf(fid,'\n');
for i = 1:r,
vx = num2str(T(i));
fprintf(fid,vx);
fprintf(fid,'\n');
end
fprintf(fid,tx);
fprintf(fid,'\n');
for i = 1:3,
Aux(1) = Cp(i);
x(i) = Cp(i);
Aux(2) = Fp(i);
y(i) = Fp(i);
disp(Aux)
nx1 = num2str(Aux(1));
fprintf(fid,nx1);
fprintf(fid,bx);
nx2 = num2str(Aux(2));
fprintf(fid,nx2);
fprintf(fid,'\n');
end
disp('*******************************************************')
figure(q);
colormap(pink);
set(q,'DefaultTextColor','cyan');
```

```
plot(x,y,'r-',x,y,'g*');
title('BUSQUEDA DE ANOMALIA DE BELADY EN PAGINACION FIFO');
ylabel('N° de Fallos de Páginas');
xlabel('N° de Celdas de Página');
disp('****************************************************')
if Fp(3) > Fp(2) | Fp(2) > Fp(1)
disp('SI se ha producido la Anomalía de Belady en la simulación efectuada.')
tx = 'SI se ha producido la Anomalía de Belady en la simulación efectuada.';
fprintf(fid,tx);
fprintf(fid,'\n');
else
disp('NO se ha producido la Anomalía de Belady en la simulación efectuada.')
tx = 'NO se ha producido la Anomalía de Belady en la simulación efectuada.';
fprintf(fid,tx);
fprintf(fid,'\n');
end
end
disp('****************************************************')
end
end
end
fclose('all');
```

21.6 Datos y Ejecuciones

Los resultados detallados de las ejecuciones se muestran paso a paso en pantalla y son grabados en el archivo de datos y ejecuciones, que en este caso ha sido el archivo **Anomalia.txt,** que es un archivo con registros de longitud variable, cuyos datos formateados, para algunas ejecuciones tomadas como referencia, se encuentran en el archivo **Anomalia.dat**.

El archivo generado (**Anomalia.txt)**, posee la siguiente estructura:

- Vector de requerimientos de páginas para la simulación.

- Pares celdas de página disponibles y fallos de páginas ocurridos.
- Conclusión respecto de la ocurrencia o no de la Anomalía de Belady en la simulación efectuada.

El contenido del mencionado archivo **Anomalia.dat** luego de varias de las ejecuciones de evaluación puede consultarse en los Anexos; un detalle parcial es el siguiente:

Vector de requerimientos de páginas para la simulación:

622 684 182 511 999 231 233 655 611 501 3 281 558 359 421 446 12 985 980 847 11 866 957 967 224 990 516 267 982 487 924 629 308 803 134 573 175 802 399 982 871 11 492 676 578 17 453 985 781 1000 389 972 228 770 221 956 498 941 155 457 842 976 30 112 409 700 547 68 886 737 949 985 621 500 623 703 765 891 577 633 535 285 360 178 365 848 543 264 497 447

Pares celdas de página disponibles y fallos de páginas ocurridos:

8 90

10 90

12 89

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

471 590 792 692 95 212 714 16 192 330 16 255 483 891 675 736 874 893 584 552 3 811 238 435 584 240 629 792 506 334 401 419 899 333 801 875 745 932 209 533 278 199 803 788 178 353 668 61 174 499 771 74 944 280 601 843 120 492 344 645 570 49 648 148 390 856 204 604 10 975 447 274 138 932 53 104 588 470 16 879 449 797 104 631 742 553 405 16 753 664

Pares celdas de página disponibles y fallos de páginas ocurridos:

8 86

10 86

12 86

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

444 876 979 864 979 26 827 777 286

Pares celdas de página disponibles y fallos de páginas ocurridos:

2 8

3 8

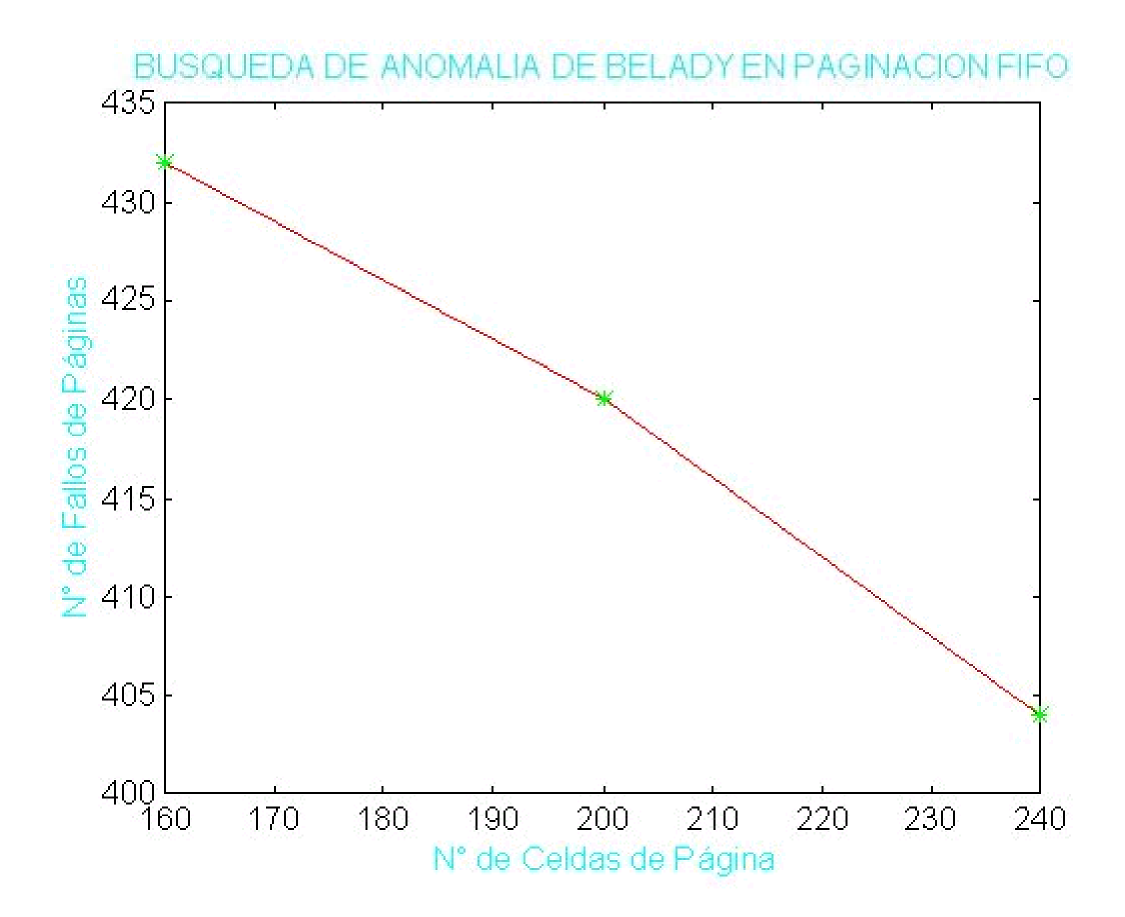


Figura 21.1: Búsqueda de Anomalía de Belady en paginación FIFO.

4 8

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

553 865 706 923 273 591 822 483 706

Pares celdas de página disponibles y fallos de páginas ocurridos:

2 9

3 9

4 9

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Uno de los gráficos producidos para una de las ejecuciones de evaluación se muestra en la Figura 21.1 de la página 568.

21.7 Resultados y Conclusiones

La utilización de *Matlab* para la resolución del problema planteado ha resultado muy satisfactoria, destacándose las facilidades y potencia del producto, en especial para el cálculo numérico.

Los *resultados obtenidos ratifican*, como era de esperarse, las *previsiones teóricas* en cuanto a que el fenómeno llamado *Anomalía de Belady* o *Anomalía FIFO* es realmente una situación muy especial que no se ha observado en ninguna de las numerosas simulaciones efectuadas; este hecho ratifica la intuición en el sentido de que si se aumenta el número de celdas de página disponibles debería disminuir el número de fallos (carencias) de páginas en un esquema de paginación FIFO.

Asimismo es necesario señalar que si bien *no se ha presentado en ningún caso de los simulados la mencionada Anomalía*, sí se han observado casos cercanos a ella, es decir casos en los que *incrementando el número de celdas de página disponibles, no se reduce el número de fallos de página*, sino que *se mantiene igual*, lo cual no llega a configurar un caso "anómalo", pero sí un caso *"llamativo"*, pues no se produce una disminución de los fallos de página registrados (para producirse la anomalía no es suficiente con que el número de fallos de página se mantenga, sino que éste debe aumentar al aumentar el número de celdas de página disponibles en la memoria principal). Estos casos "llamativos" se han observado en simulaciones con un número muy escaso de celdas de página, es decir en casos alejados de situaciones reales, no habiéndose observado en casos más cercanos a la realidad, es decir en simulaciones con centenas o miles de celdas de página disponibles.

Parte IV

Anexos

Capítulo 22

Algoritmos de Planificación del Procesador con P.O.O.

22.1 Datos y Ejecuciones

Contenido del archivo **Procesos.txt**:

****

Simulación FIFO con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 1 cambios de contexto.

****

Simulación RR - Round Robin con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación HRN con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 2 cambios de contexto.

****

Simulación RNM con 3 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación FIFO con 100 ciclos de control.

Todos los procesos han finalizado en 14 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 10 cambios de contexto.

****

Simulación RR - Round Robin con 100 ciclos de control.

Todos los procesos han finalizado en 11 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 11 cambios de contexto.

****

Simulación HRN con 100 ciclos de control.

Todos los procesos han finalizado en 26 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 15 cambios de contexto.

****

Simulación RNM con 100 ciclos de control.

Todos los procesos han finalizado en 10 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 10 cambios de contexto.

****

Simulación FIFO con 90 ciclos de control.

Todos los procesos han finalizado en 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 7 cambios de contexto.

****

Simulación RR - Round Robin con 90 ciclos de control.

Todos los procesos han finalizado en 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 9 cambios de contexto.

****

Simulación HRN con 90 ciclos de control.

Todos los procesos han finalizado en 22 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 13 cambios de contexto.

****

Simulación RNM con 90 ciclos de control.

Todos los procesos han finalizado en 14 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 14 cambios de contexto.

****

Simulación FIFO con 90 ciclos de control.

Todos los procesos han finalizado en 15 ciclos de control.

Se han simulado 6 procesos concurrentes.

Se han producido 11 cambios de contexto.

****

Simulación RR - Round Robin con 90 ciclos de control.

Todos los procesos han finalizado en 32 ciclos de control.

Se han simulado 6 procesos concurrentes.

Se han producido 32 cambios de contexto.

****

Simulación HRN con 90 ciclos de control.

Todos los procesos han finalizado en 20 ciclos de control.

Se han simulado 6 procesos concurrentes.

Se han producido 13 cambios de contexto.

****

Simulación RNM con 90 ciclos de control.

Todos los procesos han finalizado en 12 ciclos de control.

Se han simulado 6 procesos concurrentes.

Se han producido 12 cambios de contexto.

****

Simulación FIFO con 70 ciclos de control.

Todos los procesos han finalizado en 14 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 10 cambios de contexto.

****

Simulación RR - Round Robin con 70 ciclos de control.

Todos los procesos han finalizado en 11 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 11 cambios de contexto.

****

Simulación HRN con 70 ciclos de control.

Todos los procesos han finalizado en 26 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 15 cambios de contexto.

****

Simulación RNM con 70 ciclos de control.

Todos los procesos han finalizado en 10 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 10 cambios de contexto.

****

Simulación FIFO con 80 ciclos de control.

Todos los procesos han finalizado en 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 7 cambios de contexto.

****

Simulación RR - Round Robin con 80 ciclos de control.

Todos los procesos han finalizado en 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 9 cambios de contexto.

****

Simulación HRN con 80 ciclos de control.

Todos los procesos han finalizado en 22 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 13 cambios de contexto.

****

Simulación RNM con 80 ciclos de control.

Todos los procesos han finalizado en 14 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 14 cambios de contexto.

****

Simulación FIFO con 5 ciclos de control.

Se han simulado 20 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación RR - Round Robin con 5 ciclos de control.

Se han simulado 20 procesos concurrentes.

Se han producido 5 cambios de contexto.

****

Simulación HRN con 5 ciclos de control.

Se han simulado 20 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación RNM con 5 ciclos de control.

Se han simulado 20 procesos concurrentes.

Se han producido 5 cambios de contexto.

****

Simulación FIFO con 3 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 1 cambios de contexto.

****

Simulación RR - Round Robin con 3 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación HRN con 3 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 2 cambios de contexto.

****

Simulación RNM con 3 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 3 cambios de contexto.

****

Simulación FIFO con 6 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 4 cambios de contexto.

****

Simulación RR - Round Robin con 6 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 6 cambios de contexto.

****

Simulación HRN con 6 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 5 cambios de contexto.

****

Simulación RNM con 6 ciclos de control.

Se han simulado 5 procesos concurrentes.

Se han producido 6 cambios de contexto.

****

Simulación FIFO con 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 7 cambios de contexto.

****

Simulación RR - Round Robin con 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 9 cambios de contexto.

****

Simulación HRN con 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

Se han producido 6 cambios de contexto.

****

Simulación RNM con 10 ciclos de control.

Se han simulado 3 procesos concurrentes.

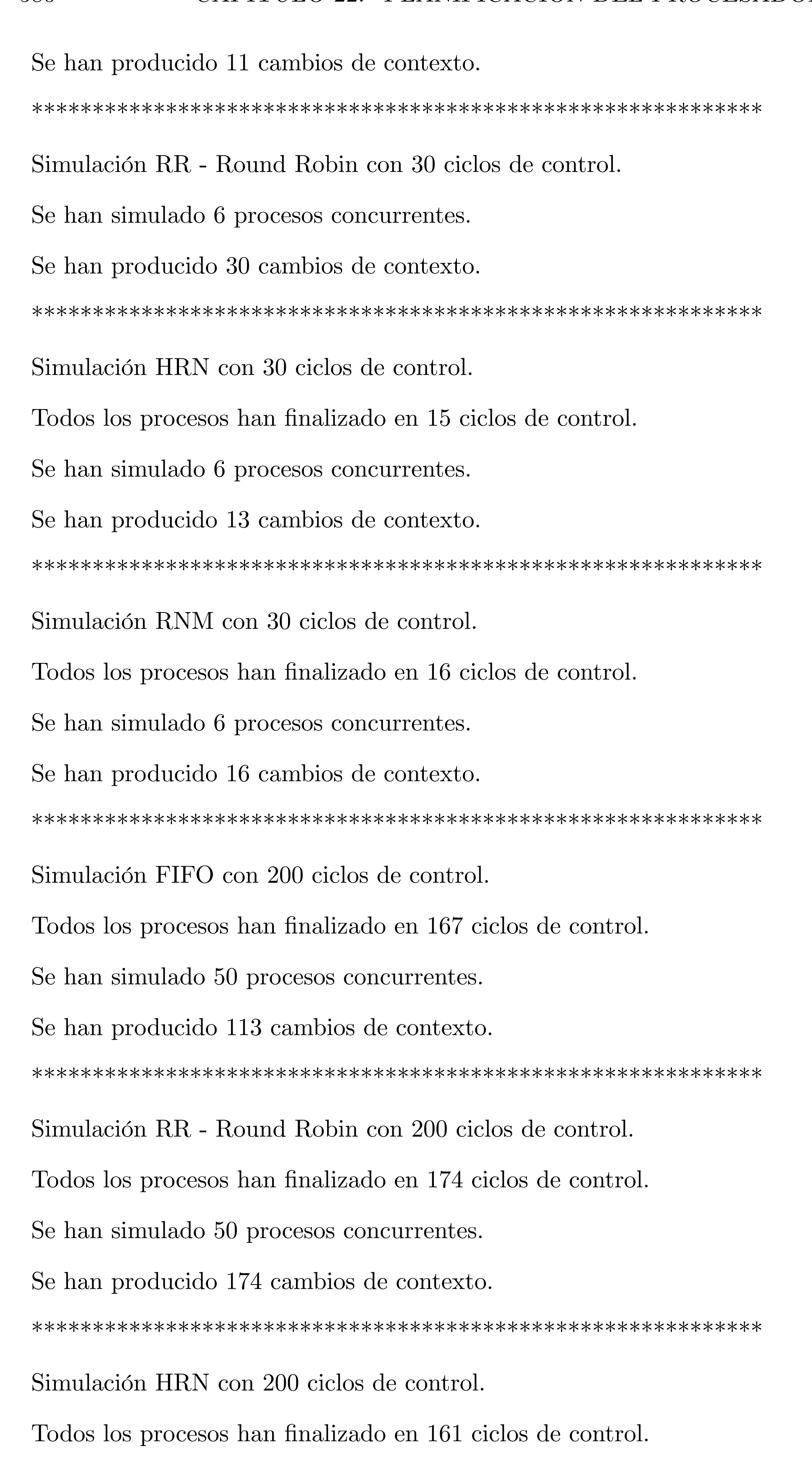
Se han producido 10 cambios de contexto.

****

Simulación FIFO con 30 ciclos de control.

Todos los procesos han finalizado en 15 ciclos de control.

Se han simulado 6 procesos concurrentes.



Se han producido 11 cambios de contexto.

****

Simulación RR - Round Robin con 30 ciclos de control.

Se han simulado 6 procesos concurrentes.

Se han producido 30 cambios de contexto.

****

Simulación HRN con 30 ciclos de control.

Todos los procesos han finalizado en 15 ciclos de control.

Se han simulado 6 procesos concurrentes.

Se han producido 13 cambios de contexto.

****

Simulación RNM con 30 ciclos de control.

Todos los procesos han finalizado en 16 ciclos de control.

Se han simulado 6 procesos concurrentes.

Se han producido 16 cambios de contexto.

****

Simulación FIFO con 200 ciclos de control.

Todos los procesos han finalizado en 167 ciclos de control.

Se han simulado 50 procesos concurrentes.

Se han producido 113 cambios de contexto.

****

Simulación RR - Round Robin con 200 ciclos de control.

Todos los procesos han finalizado en 174 ciclos de control.

Se han simulado 50 procesos concurrentes.

Se han producido 174 cambios de contexto.

****

Simulación HRN con 200 ciclos de control.

Todos los procesos han finalizado en 161 ciclos de control.

Se han simulado 50 procesos concurrentes.

Se han producido 98 cambios de contexto.

****

Simulación RNM con 200 ciclos de control.

Todos los procesos han finalizado en 198 ciclos de control.

Se han simulado 50 procesos concurrentes.

Se han producido 198 cambios de contexto.

****

Simulación FIFO con 30 ciclos de control.

Todos los procesos han finalizado en 25 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 19 cambios de contexto.

****

Simulación RR - Round Robin con 30 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 30 cambios de contexto.

****

Simulación HRN con 30 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 20 cambios de contexto.

****

Simulación RNM con 30 ciclos de control.

Se han simulado 10 procesos concurrentes.

Se han producido 30 cambios de contexto.

****

Simulación FIFO con 50 ciclos de control.

Todos los procesos han finalizado en 47 ciclos de control.

Se han simulado 15 procesos concurrentes.

Se han producido 33 cambios de contexto.

****

Simulación RR - Round Robin con 50 ciclos de control.

Se han simulado 15 procesos concurrentes.

Se han producido 50 cambios de contexto.

****

Simulación HRN con 50 ciclos de control.

Se han simulado 15 procesos concurrentes.

Se han producido 31 cambios de contexto.

****

Simulación RNM con 50 ciclos de control.

Todos los procesos han finalizado en 46 ciclos de control.

Se han simulado 15 procesos concurrentes.

Se han producido 46 cambios de contexto.

****

Simulación FIFO con 60 ciclos de control.

Todos los procesos han finalizado en 17 ciclos de control.

Se han simulado 8 procesos concurrentes.

Se han producido 13 cambios de contexto.

****

Simulación RR - Round Robin con 60 ciclos de control.

Todos los procesos han finalizado en 32 ciclos de control.

Se han simulado 8 procesos concurrentes.

Se han producido 32 cambios de contexto.

****

Simulación HRN con 60 ciclos de control.

Todos los procesos han finalizado en 29 ciclos de control.

Se han simulado 8 procesos concurrentes.

Se han producido 19 cambios de contexto.

****

Simulación RNM con 60 ciclos de control.

Todos los procesos han finalizado en 28 ciclos de control.

Se han simulado 8 procesos concurrentes.

Se han producido 28 cambios de contexto.

Capítulo 23

Paginación de Memoria Virtual con Sistemas Expertos

23.1 Programas Desarrollados y Datos y Ejecuciones

Contenido del archivo *"paginas.ma":*

```
(* Ejemplo de Sistema Experto de paginación en memoria virtual *)
Paginacion[Prior_, Verosimilitudes_, DatosPagina_] :=
Module[{Prior1 = Prior, Posterior, i, j, aux, aux1,
n = Length[Prior], m = Length[DatosPagina]},
Print["******************************************"];
Print["* Análisis del encuadre de la página *"];
Print["* según las estrategias de paginación, *"];
Print["* lo que determinará la posibilidad de ser *"];
Print["* desalojada de la memoria principal *"];
Print["* según las distintas estrategias. *"];
Print["******************************************"];
Do[
Posterior = {};
Do[AppendTo[Posterior, If[DatosPagina[[j]] > 0,
(Verosimilitudes[i, Abs[DatosPagina[[j]]]]) * Prior1[[i]],
(1.0 - Verosimilitudes[[i, Abs[DatosPagina[[j]]]]])
```

```
* Prior1[[i]]]]
, {i, 1, n}];
aux = Sum[Posterior[[i]], {i, 1, n}];
Prior1 = Posterior / aux;
aux1 = Table[{Prior1[[i]], Estrategias[[i]]}, {i, 1, n}];
aux1 = Sort[aux1];
Print[" "]; Print[" "];
Print["Resultados tras la consideración de la pauta:"];
Print[" ", Pautas[[Abs[DatosPagina[[j]]]]]];
Print[" "];
Do[Print[aux1[[i, 2]], ": ", aux1[[i, 1]]]
, {i, n, 1, -1}];
, {j, 1, m}];
Print[" "]; Print[" "];
Print["**************************************************"];
Print["Resultado final:"];
Print[" La página tiene las siguientes probabilidades"];
Print[" de ser removida de la memoria principal según"];
Print[" las estrategias consideradas:"];
Print[" "];
Do[Print[aux1[[i, 2]], ": ", aux1[[i, 1]]]
, {i, n, 1, -1}];
Print["**************************************************"];
]
Estrategias = {"Tiempo estimado de no utilización mayor",
"Reposición FIFO - Tiempo mayor en memoria principal",
"Reposición LRU - Página menos recientemente usada",
"Reposición LFU - Página menos frecuentemente usada",
```

"Reposición NUR - Página no usada recientemente"};

Pautas = {"Tiempo hasta la próxima utilización: - 5.000 miliseg.",

"Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.",

"Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.",

"Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.",

"Tiempo hasta la próxima utilización: + 40.000 miliseg.",

"Tiempo en mem. ppal.: - 1.000 miliseg.",

"Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.",

"Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.",

"Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.",

"Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.",

"Tiempo en mem. ppal.: + 30.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.",

"Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.",

"Intensidad de utilización de la página: Menos de 50 veces",

"Intensidad de utilización de la página: Entre 50 y 100 veces",

"Intensidad de utilización de la página: Entre 101 y 200 veces",

"Intensidad de utilización de la página: Entre 201 y 300 veces",

"Intensidad de utilización de la página: Más de 300 veces",

"Página no referenciada",

"Página sí referenciada y no modificada",

”Página sí referenciada y sí modificada”};

Prior = {0.1, 0.2, 0.25, 0.3, 0.5};

Verosimilitudes =

{{0.15, 0.25, 0.40, 0.60, 0.95,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.10, 0.20, 0.40, 0.60, 0.80, 1.00,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.85, 1.00,

0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

1.00, 0.70, 0.50, 0.25, 0.05,

0.01, 0.01, 0.01},

{0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.01, 0.01, 0.01, 0.01,

1.00, 0.50, 0.05}};

(* Ejemplo de página con mucho tiempo *)

(* estimado hasta la próxima utilización *)

DatosPagina =

{-1, -2, -3, -4, 5,

-6, -7, -8, -9, -10, -11,

-12, -13, -14, -15, -16, -17, -18, -19, -20,

-21, -22, -23, -24, -25,

-26, -27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.394967

Reposición LFU - Página menos frecuentemente usada: 0.23698

Reposición LRU - Página menos recientemente usada: 0.197484

Reposición FIFO - Tiempo mayor en memoria principal: 0.157987

Tiempo estimado de no utilización mayor: 0.0125818

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.547613

Reposición NUR - Página no usada recientemente: 0.180955

Reposición LFU - Página menos frecuentemente usada: 0.108573

Reposición LRU - Página menos recientemente usada: 0.0904773

Reposición FIFO - Tiempo mayor en memoria principal: 0.0723819

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Tiempo estimado de no utilización mayor: 0.551241

Reposición NUR - Página no usada recientemente: 0.182153

Reposición LFU - Página menos frecuentemente usada: 0.109292

Reposición LRU - Página menos recientemente usada: 0.0910766

Reposición FIFO - Tiempo mayor en memoria principal: 0.0662375

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Tiempo estimado de no utilización mayor: 0.558338

Reposición NUR - Página no usada recientemente: 0.184499

Reposición LFU - Página menos frecuentemente usada: 0.110699

Reposición LRU - Página menos recientemente usada: 0.0922493

Reposición FIFO - Tiempo mayor en memoria principal: 0.0542145

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Tiempo estimado de no utilización mayor: 0.570523

Reposición NUR - Página no usada recientemente: 0.188525

Reposición LFU - Página menos frecuentemente usada: 0.113115

Reposición LRU - Página menos recientemente usada: 0.0942625

Reposición FIFO - Tiempo mayor en memoria principal: 0.0335743

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Tiempo estimado de no utilización mayor: 0.582172

Reposición NUR - Página no usada recientemente: 0.192374

Reposición LFU - Página menos frecuentemente usada: 0.115425

Reposición LRU - Página menos recientemente usada: 0.0961871

Reposición FIFO - Tiempo mayor en memoria principal: 0.0138424

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Tiempo estimado de no utilización mayor: 0.588674

Reposición NUR - Página no usada recientemente: 0.194523

Reposición LFU - Página menos frecuentemente usada: 0.116714

Reposición LRU - Página menos recientemente usada: 0.0972615

Reposición FIFO - Tiempo mayor en memoria principal: 0.00282767

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Tiempo estimado de no utilización mayor: 0.590344

Reposición NUR - Página no usada recientemente: 0.195075

Reposición LFU - Página menos frecuentemente usada: 0.117045

Reposición LRU - Página menos recientemente usada: 0.0975373

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Tiempo estimado de no utilización mayor: 0.595625

Reposición NUR - Página no usada recientemente: 0.19682

Reposición LFU - Página menos frecuentemente usada: 0.118092

Reposición LRU - Página menos recientemente usada: 0.0894635

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Tiempo estimado de no utilización mayor: 0.60603

Reposición NUR - Página no usada recientemente: 0.200258

Reposición LFU - Página menos frecuentemente usada: 0.120155

Reposición LRU - Página menos recientemente usada: 0.0735567

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Tiempo estimado de no utilización mayor: 0.619376

Reposición NUR - Página no usada recientemente: 0.204668

Reposición LFU - Página menos frecuentemente usada: 0.122801

Reposición LRU - Página menos recientemente usada: 0.0531551

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Tiempo estimado de no utilización mayor: 0.632623

Reposición NUR - Página no usada recientemente: 0.209045

Reposición LFU - Página menos frecuentemente usada: 0.125427

Reposición LRU - Página menos recientemente usada: 0.0329042

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Tiempo estimado de no utilización mayor: 0.643096

Reposición NUR - Página no usada recientemente: 0.212506

Reposición LFU - Página menos frecuentemente usada: 0.127504

Reposición LRU - Página menos recientemente usada: 0.0168934

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Tiempo estimado de no utilización mayor: 0.649637

Reposición NUR - Página no usada recientemente: 0.214668

Reposición LFU - Página menos frecuentemente usada: 0.128801

Reposición LRU - Página menos recientemente usada: 0.00689504

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Tiempo estimado de no utilización mayor: 0.652774

Reposición NUR - Página no usada recientemente: 0.215704

Reposición LFU - Página menos frecuentemente usada: 0.129422

Reposición LRU - Página menos recientemente usada: 0.0020995

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.653939

Reposición NUR - Página no usada recientemente: 0.216089

Reposición LFU - Página menos frecuentemente usada: 0.129653

Reposición LRU - Página menos recientemente usada: 0.000318673

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.654147

Reposición NUR - Página no usada recientemente: 0.216158

Reposición LFU - Página menos frecuentemente usada: 0.129695

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Tiempo estimado de no utilización mayor: 0.75163

Reposición NUR - Página no usada recientemente: 0.24837

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

(* Ejemplo de página con mucho tiempo *)

(* de permanencia en memoria principal *)

DatosPagina =

{-1, -2, -3, -4, -5,

-6, -7, -8, -9, -10, 11,

-12, -13, -14, -15, -16, -17, -18, -19, -20,

-21, -22, -23, -24, -25,

-26, -27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.394967

Reposición LFU - Página menos frecuentemente usada: 0.23698

Reposición LRU - Página menos recientemente usada: 0.197484

Reposición FIFO - Tiempo mayor en memoria principal: 0.157987

Tiempo estimado de no utilización mayor: 0.0125818

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.399743

Reposición LFU - Página menos frecuentemente usada: 0.239846

Reposición LRU - Página menos recientemente usada: 0.199871

Reposición FIFO - Tiempo mayor en memoria principal: 0.159897

Tiempo estimado de no utilización mayor: 0.000643126

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.405639

Reposición LFU - Página menos frecuentemente usada: 0.243383

Reposición LRU - Página menos recientemente usada: 0.20282

Reposición FIFO - Tiempo mayor en memoria principal: 0.147505

Tiempo estimado de no utilización mayor: 0.000652612

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.417457

Reposición LFU - Página menos frecuentemente usada: 0.250474

Reposición LRU - Página menos recientemente usada: 0.208728

Reposición FIFO - Tiempo mayor en memoria principal: 0.122669

Tiempo estimado de no utilización mayor: 0.000671626

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.438655

Reposición LFU - Página menos frecuentemente usada: 0.263193

Reposición LRU - Página menos recientemente usada: 0.219327

Reposición FIFO - Tiempo mayor en memoria principal: 0.0781197

Tiempo estimado de no utilización mayor: 0.000705729

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.460074

Reposición LFU - Página menos frecuentemente usada: 0.276044

Reposición LRU - Página menos recientemente usada: 0.230037

Reposición FIFO - Tiempo mayor en memoria principal: 0.0331048

Tiempo estimado de no utilización mayor: 0.00074019

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.472557

Reposición LFU - Página menos frecuentemente usada: 0.283534

Reposición LRU - Página menos recientemente usada: 0.236279

Reposición FIFO - Tiempo mayor en memoria principal: 0.0068693

Tiempo estimado de no utilización mayor: 0.000760274

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.408872

Reposición NUR - Página no usada recientemente: 0.281274

Reposición LFU - Página menos frecuentemente usada: 0.168764

Reposición LRU - Página menos recientemente usada: 0.140637

Tiempo estimado de no utilización mayor: 0.000452528

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.414167

Reposición NUR - Página no usada recientemente: 0.284917

Reposición LFU - Página menos frecuentemente usada: 0.17095

Reposición LRU - Página menos recientemente usada: 0.129508

Tiempo estimado de no utilización mayor: 0.000458388

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.424724

Reposición NUR - Página no usada recientemente: 0.292179

Reposición LFU - Página menos frecuentemente usada: 0.175307

Reposición LRU - Página menos recientemente usada: 0.10732

Tiempo estimado de no utilización mayor: 0.000470072

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.438509

Reposición NUR - Página no usada recientemente: 0.301662

Reposición LFU - Página menos frecuentemente usada: 0.180997

Reposición LRU - Página menos recientemente usada: 0.0783458

Tiempo estimado de no utilización mayor: 0.000485329

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.452474

Reposición NUR - Página no usada recientemente: 0.311269

Reposición LFU - Página menos frecuentemente usada: 0.186761

Reposición LRU - Página menos recientemente usada: 0.0489945

Tiempo estimado de no utilización mayor: 0.000500785

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.463719

Reposición NUR - Página no usada recientemente: 0.319005

Reposición LFU - Página menos frecuentemente usada: 0.191403

Reposición LRU - Página menos recientemente usada: 0.0253596

Tiempo estimado de no utilización mayor: 0.000513231

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.470835

Reposición NUR - Página no usada recientemente: 0.3239

Reposición LFU - Página menos frecuentemente usada: 0.19434

Reposición LRU - Página menos recientemente usada: 0.0104036

Tiempo estimado de no utilización mayor: 0.000521107

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.474274

Reposición NUR - Página no usada recientemente: 0.326266

Reposición LFU - Página menos frecuentemente usada: 0.19576

Reposición LRU - Página menos recientemente usada: 0.00317562

Tiempo estimado de no utilización mayor: 0.000524913

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.475555

Reposición NUR - Página no usada recientemente: 0.327147

Reposición LFU - Página menos frecuentemente usada: 0.196288

Tiempo estimado de no utilización mayor: 0.000526331

Reposición LRU - Página menos recientemente usada: 0.000482454

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.475785

Reposición NUR - Página no usada recientemente: 0.327305

Reposición LFU - Página menos frecuentemente usada: 0.196383

Tiempo estimado de no utilización mayor: 0.000526585

Reposición LRU - Página menos recientemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.592055

Reposición NUR - Página no usada recientemente: 0.40729

Tiempo estimado de no utilización mayor: 0.000655269

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.592055

Reposición NUR - Página no usada recientemente: 0.40729

Tiempo estimado de no utilización mayor: 0.000655269

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.592055

Reposición NUR - Página no usada recientemente: 0.40729

Tiempo estimado de no utilización mayor: 0.000655269

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.592055

Reposición NUR - Página no usada recientemente: 0.40729

Tiempo estimado de no utilización mayor: 0.000655269

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.592055

Reposición NUR - Página no usada recientemente: 0.40729

Tiempo estimado de no utilización mayor: 0.000655269

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Reposición FIFO - Tiempo mayor en memoria principal: 0.998894

Tiempo estimado de no utilización mayor: 0.00110555

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Reposición FIFO - Tiempo mayor en memoria principal: 0.998894

Tiempo estimado de no utilización mayor: 0.00110555

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Reposición FIFO - Tiempo mayor en memoria principal: 0.998894

Tiempo estimado de no utilización mayor: 0.00110555

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Reposición FIFO - Tiempo mayor en memoria principal: 0.998894

Tiempo estimado de no utilización mayor: 0.00110555

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

****

(* Ejemplo de página con mucho tiempo *)

(* de permanencia en memoria principal sin uso *)

DatosPagina =

{-1, -2, -3, -4, -5,

-6, -7, -8, -9, -10, -11,

-12, -13, -14, -15, -16, -17, -18, -19, 20,

-21, -22, -23, -24, -25,

-26, -27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.394967

Reposición LFU - Página menos frecuentemente usada: 0.23698

Reposición LRU - Página menos recientemente usada: 0.197484

Reposición FIFO - Tiempo mayor en memoria principal: 0.157987

Tiempo estimado de no utilización mayor: 0.0125818

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.399743

Reposición LFU - Página menos frecuentemente usada: 0.239846

Reposición LRU - Página menos recientemente usada: 0.199871

Reposición FIFO - Tiempo mayor en memoria principal: 0.159897

Tiempo estimado de no utilización mayor: 0.000643126

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.405639

Reposición LFU - Página menos frecuentemente usada: 0.243383

Reposición LRU - Página menos recientemente usada: 0.20282

Reposición FIFO - Tiempo mayor en memoria principal: 0.147505

Tiempo estimado de no utilización mayor: 0.000652612

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.417457

Reposición LFU - Página menos frecuentemente usada: 0.250474

Reposición LRU - Página menos recientemente usada: 0.208728

Reposición FIFO - Tiempo mayor en memoria principal: 0.122669

Tiempo estimado de no utilización mayor: 0.000671626

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.438655

Reposición LFU - Página menos frecuentemente usada: 0.263193

Reposición LRU - Página menos recientemente usada: 0.219327

Reposición FIFO - Tiempo mayor en memoria principal: 0.0781197

Tiempo estimado de no utilización mayor: 0.000705729

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.460074

Reposición LFU - Página menos frecuentemente usada: 0.276044

Reposición LRU - Página menos recientemente usada: 0.230037

Reposición FIFO - Tiempo mayor en memoria principal: 0.0331048

Tiempo estimado de no utilización mayor: 0.00074019

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.472557

Reposición LFU - Página menos frecuentemente usada: 0.283534

Reposición LRU - Página menos recientemente usada: 0.236279

Reposición FIFO - Tiempo mayor en memoria principal: 0.0068693

Tiempo estimado de no utilización mayor: 0.000760274

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.475826

Reposición LFU - Página menos frecuentemente usada: 0.285496

Reposición LRU - Página menos recientemente usada: 0.237913

Tiempo estimado de no utilización mayor: 0.000765532

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.486345

Reposición LFU - Página menos frecuentemente usada: 0.291807

Reposición LRU - Página menos recientemente usada: 0.221066

Tiempo estimado de no utilización mayor: 0.000782456

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.507893

Reposición LFU - Página menos frecuentemente usada: 0.304736

Reposición LRU - Página menos recientemente usada: 0.186554

Tiempo estimado de no utilización mayor: 0.000817124

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.537252

Reposición LFU - Página menos frecuentemente usada: 0.322351

Reposición LRU - Página menos recientemente usada: 0.139532

Tiempo estimado de no utilización mayor: 0.000864358

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.568501

Reposición LFU - Página menos frecuentemente usada: 0.341101

Reposición LRU - Página menos recientemente usada: 0.0894834

Tiempo estimado de no utilización mayor: 0.000914633

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.594847

Reposición LFU - Página menos frecuentemente usada: 0.356908

Reposición LRU - Página menos recientemente usada: 0.047288

Tiempo estimado de no utilización mayor: 0.000957019

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.612097

Reposición LFU - Página menos frecuentemente usada: 0.367258

Reposición LRU - Página menos recientemente usada: 0.0196603

Tiempo estimado de no utilización mayor: 0.000984772

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.620601

Reposición LFU - Página menos frecuentemente usada: 0.37236

Reposición LRU - Página menos recientemente usada: 0.00604044

Tiempo estimado de no utilización mayor: 0.000998453

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.623798

Reposición LFU - Página menos frecuentemente usada: 0.374279

Tiempo estimado de no utilización mayor: 0.0010036

Reposición LRU - Página menos recientemente usada: 0.000919934

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.571729

Reposición LFU - Página menos frecuentemente usada: 0.343037

Reposición LRU - Página menos recientemente usada: 0.0843146

Tiempo estimado de no utilización mayor: 0.000919825

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición NUR - Página no usada recientemente: 0.87026

Reposición LRU - Página menos recientemente usada: 0.12834

Tiempo estimado de no utilización mayor: 0.00140012

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición NUR - Página no usada recientemente: 0.87026

Reposición LRU - Página menos recientemente usada: 0.12834

Tiempo estimado de no utilización mayor: 0.00140012

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición NUR - Página no usada recientemente: 0.87026

Reposición LRU - Página menos recientemente usada: 0.12834

Tiempo estimado de no utilización mayor: 0.00140012

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición NUR - Página no usada recientemente: 0.87026

Reposición LRU - Página menos recientemente usada: 0.12834

Tiempo estimado de no utilización mayor: 0.00140012

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición NUR - Página no usada recientemente: 0.87026

Reposición LRU - Página menos recientemente usada: 0.12834

Tiempo estimado de no utilización mayor: 0.00140012

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Reposición LRU - Página menos recientemente usada: 0.989208

Tiempo estimado de no utilización mayor: 0.0107917

Reposición NUR - Página no usada recientemente: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Reposición LRU - Página menos recientemente usada: 0.989208

Tiempo estimado de no utilización mayor: 0.0107917

Reposición NUR - Página no usada recientemente: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Reposición LRU - Página menos recientemente usada: 0.989208

Tiempo estimado de no utilización mayor: 0.0107917

Reposición NUR - Página no usada recientemente: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Reposición LRU - Página menos recientemente usada: 0.989208

Tiempo estimado de no utilización mayor: 0.0107917

Reposición NUR - Página no usada recientemente: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

(* Ejemplo de página poco utilizada *)

DatosPagina =

{-1, -2, -3, -4, -5,

-6, -7, -8, -9, -10, -11,

-12, -13, -14, -15, -16, -17, -18, -19, -20,

21, -22, -23, -24, -25,

-26, -27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.394967

Reposición LFU - Página menos frecuentemente usada: 0.23698

Reposición LRU - Página menos recientemente usada: 0.197484

Reposición FIFO - Tiempo mayor en memoria principal: 0.157987

Tiempo estimado de no utilización mayor: 0.0125818

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.399743

Reposición LFU - Página menos frecuentemente usada: 0.239846

Reposición LRU - Página menos recientemente usada: 0.199871

Reposición FIFO - Tiempo mayor en memoria principal: 0.159897

Tiempo estimado de no utilización mayor: 0.000643126

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.405639

Reposición LFU - Página menos frecuentemente usada: 0.243383

Reposición LRU - Página menos recientemente usada: 0.20282

Reposición FIFO - Tiempo mayor en memoria principal: 0.147505

Tiempo estimado de no utilización mayor: 0.000652612

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.417457

Reposición LFU - Página menos frecuentemente usada: 0.250474

Reposición LRU - Página menos recientemente usada: 0.208728

Reposición FIFO - Tiempo mayor en memoria principal: 0.122669

Tiempo estimado de no utilización mayor: 0.000671626

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.438655

Reposición LFU - Página menos frecuentemente usada: 0.263193

Reposición LRU - Página menos recientemente usada: 0.219327

Reposición FIFO - Tiempo mayor en memoria principal: 0.0781197

Tiempo estimado de no utilización mayor: 0.000705729

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.460074

Reposición LFU - Página menos frecuentemente usada: 0.276044

Reposición LRU - Página menos recientemente usada: 0.230037

Reposición FIFO - Tiempo mayor en memoria principal: 0.0331048

Tiempo estimado de no utilización mayor: 0.00074019

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.472557

Reposición LFU - Página menos frecuentemente usada: 0.283534

Reposición LRU - Página menos recientemente usada: 0.236279

Reposición FIFO - Tiempo mayor en memoria principal: 0.0068693

Tiempo estimado de no utilización mayor: 0.000760274

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.475826

Reposición LFU - Página menos frecuentemente usada: 0.285496

Reposición LRU - Página menos recientemente usada: 0.237913

Tiempo estimado de no utilización mayor: 0.000765532

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.486345

Reposición LFU - Página menos frecuentemente usada: 0.291807

Reposición LRU - Página menos recientemente usada: 0.221066

Tiempo estimado de no utilización mayor: 0.000782456

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.507893

Reposición LFU - Página menos frecuentemente usada: 0.304736

Reposición LRU - Página menos recientemente usada: 0.186554

Tiempo estimado de no utilización mayor: 0.000817124

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.537252

Reposición LFU - Página menos frecuentemente usada: 0.322351

Reposición LRU - Página menos recientemente usada: 0.139532

Tiempo estimado de no utilización mayor: 0.000864358

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.568501

Reposición LFU - Página menos frecuentemente usada: 0.341101

Reposición LRU - Página menos recientemente usada: 0.0894834

Tiempo estimado de no utilización mayor: 0.000914633

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.594847

Reposición LFU - Página menos frecuentemente usada: 0.356908

Reposición LRU - Página menos recientemente usada: 0.047288

Tiempo estimado de no utilización mayor: 0.000957019

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.612097

Reposición LFU - Página menos frecuentemente usada: 0.367258

Reposición LRU - Página menos recientemente usada: 0.0196603

Tiempo estimado de no utilización mayor: 0.000984772

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.620601

Reposición LFU - Página menos frecuentemente usada: 0.37236

Reposición LRU - Página menos recientemente usada: 0.00604044

Tiempo estimado de no utilización mayor: 0.000998453

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.623798

Reposición LFU - Página menos frecuentemente usada: 0.374279

Tiempo estimado de no utilización mayor: 0.0010036

Reposición LRU - Página menos recientemente usada: 0.000919934

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.624372

Reposición LFU - Página menos frecuentemente usada: 0.374623

Tiempo estimado de no utilización mayor: 0.00100452

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición LFU - Página menos frecuentemente usada: 0.983581

Reposición NUR - Página no usada recientemente: 0.016393

Tiempo estimado de no utilización mayor: 0.0000263739

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición LFU - Página menos frecuentemente usada: 0.947788

Reposición NUR - Página no usada recientemente: 0.0521283

Tiempo estimado de no utilización mayor: 0.0000838666

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición LFU - Página menos frecuentemente usada: 0.901652

Reposición NUR - Página no usada recientemente: 0.0981899

Tiempo estimado de no utilización mayor: 0.000157973

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición LFU - Página menos frecuentemente usada: 0.874142

Reposición NUR - Página no usada recientemente: 0.125656

Tiempo estimado de no utilización mayor: 0.000202162

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición LFU - Página menos frecuentemente usada: 0.869534

Reposición NUR - Página no usada recientemente: 0.130257

Tiempo estimado de no utilización mayor: 0.000209563

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Reposición LFU - Página menos frecuentemente usada: 0.999759

Tiempo estimado de no utilización mayor: 0.000240948

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Reposición LFU - Página menos frecuentemente usada: 0.999759

Tiempo estimado de no utilización mayor: 0.000240948

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Reposición LFU - Página menos frecuentemente usada: 0.999759

Tiempo estimado de no utilización mayor: 0.000240948

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Reposición LFU - Página menos frecuentemente usada: 0.999759

Tiempo estimado de no utilización mayor: 0.000240948

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

(* Ejemplo de página no referenciada *)

DatosPagina =

{-1, -2, -3, -4, -5,

-6, -7, -8, -9, -10, -11,

-12, -13, -14, -15, -16, -17, -18, -19, -20,

-21, -22, -23, -24, -25,

26, -27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.394967

Reposición LFU - Página menos frecuentemente usada: 0.23698

Reposición LRU - Página menos recientemente usada: 0.197484

Reposición FIFO - Tiempo mayor en memoria principal: 0.157987

Tiempo estimado de no utilización mayor: 0.0125818

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.399743

Reposición LFU - Página menos frecuentemente usada: 0.239846

Reposición LRU - Página menos recientemente usada: 0.199871

Reposición FIFO - Tiempo mayor en memoria principal: 0.159897

Tiempo estimado de no utilización mayor: 0.000643126

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.405639

Reposición LFU - Página menos frecuentemente usada: 0.243383

Reposición LRU - Página menos recientemente usada: 0.20282

Reposición FIFO - Tiempo mayor en memoria principal: 0.147505

Tiempo estimado de no utilización mayor: 0.000652612

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.417457

Reposición LFU - Página menos frecuentemente usada: 0.250474

Reposición LRU - Página menos recientemente usada: 0.208728

Reposición FIFO - Tiempo mayor en memoria principal: 0.122669

Tiempo estimado de no utilización mayor: 0.000671626

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.438655

Reposición LFU - Página menos frecuentemente usada: 0.263193

Reposición LRU - Página menos recientemente usada: 0.219327

Reposición FIFO - Tiempo mayor en memoria principal: 0.0781197

Tiempo estimado de no utilización mayor: 0.000705729

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.460074

Reposición LFU - Página menos frecuentemente usada: 0.276044

Reposición LRU - Página menos recientemente usada: 0.230037

Reposición FIFO - Tiempo mayor en memoria principal: 0.0331048

Tiempo estimado de no utilización mayor: 0.00074019

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.472557

Reposición LFU - Página menos frecuentemente usada: 0.283534

Reposición LRU - Página menos recientemente usada: 0.236279

Reposición FIFO - Tiempo mayor en memoria principal: 0.0068693

Tiempo estimado de no utilización mayor: 0.000760274

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.475826

Reposición LFU - Página menos frecuentemente usada: 0.285496

Reposición LRU - Página menos recientemente usada: 0.237913

Tiempo estimado de no utilización mayor: 0.000765532

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.486345

Reposición LFU - Página menos frecuentemente usada: 0.291807

Reposición LRU - Página menos recientemente usada: 0.221066

Tiempo estimado de no utilización mayor: 0.000782456

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.507893

Reposición LFU - Página menos frecuentemente usada: 0.304736

Reposición LRU - Página menos recientemente usada: 0.186554

Tiempo estimado de no utilización mayor: 0.000817124

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.537252

Reposición LFU - Página menos frecuentemente usada: 0.322351

Reposición LRU - Página menos recientemente usada: 0.139532

Tiempo estimado de no utilización mayor: 0.000864358

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.568501

Reposición LFU - Página menos frecuentemente usada: 0.341101

Reposición LRU - Página menos recientemente usada: 0.0894834

Tiempo estimado de no utilización mayor: 0.000914633

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.594847

Reposición LFU - Página menos frecuentemente usada: 0.356908

Reposición LRU - Página menos recientemente usada: 0.047288

Tiempo estimado de no utilización mayor: 0.000957019

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.612097

Reposición LFU - Página menos frecuentemente usada: 0.367258

Reposición LRU - Página menos recientemente usada: 0.0196603

Tiempo estimado de no utilización mayor: 0.000984772

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.620601

Reposición LFU - Página menos frecuentemente usada: 0.37236

Reposición LRU - Página menos recientemente usada: 0.00604044

Tiempo estimado de no utilización mayor: 0.000998453

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.623798

Reposición LFU - Página menos frecuentemente usada: 0.374279

Tiempo estimado de no utilización mayor: 0.0010036

Reposición LRU - Página menos recientemente usada: 0.000919934

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.624372

Reposición LFU - Página menos frecuentemente usada: 0.374623

Tiempo estimado de no utilización mayor: 0.00100452

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición NUR - Página no usada recientemente: 0.998394

Tiempo estimado de no utilización mayor: 0.00160627

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición NUR - Página no usada recientemente: 0.998394

Tiempo estimado de no utilización mayor: 0.00160627

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición NUR - Página no usada recientemente: 0.998394

Tiempo estimado de no utilización mayor: 0.00160627

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición NUR - Página no usada recientemente: 0.998394

Tiempo estimado de no utilización mayor: 0.00160627

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición NUR - Página no usada recientemente: 0.998394

Tiempo estimado de no utilización mayor: 0.00160627

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Reposición NUR - Página no usada recientemente: 0.999984

Tiempo estimado de no utilización mayor: 0.0000160882

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Reposición NUR - Página no usada recientemente: 0.999968

Tiempo estimado de no utilización mayor: 0.0000318542

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Reposición NUR - Página no usada recientemente: 0.999967

Tiempo estimado de no utilización mayor: 0.0000331954

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Reposición NUR - Página no usada recientemente: 0.999967

Tiempo estimado de no utilización mayor: 0.0000331954

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

(* Ejemplo de página con:*)

(* -5.000 miliseg. hasta el próximo uso *)

(* +30.000 miliseg. en memoria principal *)

(* 10.001 - 15.000 miliseg. en mem. ppal. sin uso *)

(* 101 - 200 veces de utilización *)

(* sí referenciada y sí modificada *)

DatosPagina =

{1, -2, -3, -4, -5,

-6, -7, -8, -9, -10, 11,

-12, -13, 14, -15, -16, -17, -18, -19, -20,

-21, -22, 23, -24, -25,

-26, -27, 28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Tiempo estimado de no utilización mayor: 0.545455

Reposición NUR - Página no usada recientemente: 0.181818

Reposición LFU - Página menos frecuentemente usada: 0.109091

Reposición LRU - Página menos recientemente usada: 0.0909091

Reposición FIFO - Tiempo mayor en memoria principal: 0.0727273

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Tiempo estimado de no utilización mayor: 0.47619

Reposición NUR - Página no usada recientemente: 0.209524

Reposición LFU - Página menos frecuentemente usada: 0.125714

Reposición LRU - Página menos recientemente usada: 0.104762

Reposición FIFO - Tiempo mayor en memoria principal: 0.0838095

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Tiempo estimado de no utilización mayor: 0.35524

Reposición NUR - Página no usada recientemente: 0.257904

Reposición LFU - Página menos frecuentemente usada: 0.154742

Reposición LRU - Página menos recientemente usada: 0.128952

Reposición FIFO - Tiempo mayor en memoria principal: 0.103162

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.327168

Reposición LFU - Página menos frecuentemente usada: 0.196301

Tiempo estimado de no utilización mayor: 0.182079

Reposición LRU - Página menos recientemente usada: 0.163584

Reposición FIFO - Tiempo mayor en memoria principal: 0.130867

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.395553

Reposición LFU - Página menos frecuentemente usada: 0.237332

Reposición LRU - Página menos recientemente usada: 0.197776

Reposición FIFO - Tiempo mayor en memoria principal: 0.158221

Tiempo estimado de no utilización mayor: 0.011118

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.401325

Reposición LFU - Página menos frecuentemente usada: 0.240795

Reposición LRU - Página menos recientemente usada: 0.200663

Reposición FIFO - Tiempo mayor en memoria principal: 0.145936

Tiempo estimado de no utilización mayor: 0.0112803

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.41289

Reposición LFU - Página menos frecuentemente usada: 0.247734

Reposición LRU - Página menos recientemente usada: 0.206445

Reposición FIFO - Tiempo mayor en memoria principal: 0.121327

Tiempo estimado de no utilización mayor: 0.0116053

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.433614

Reposición LFU - Página menos frecuentemente usada: 0.260169

Reposición LRU - Página menos recientemente usada: 0.216807

Reposición FIFO - Tiempo mayor en memoria principal: 0.0772221

Tiempo estimado de no utilización mayor: 0.0121878

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.454532

Reposición LFU - Página menos frecuentemente usada: 0.272719

Reposición LRU - Página menos recientemente usada: 0.227266

Reposición FIFO - Tiempo mayor en memoria principal: 0.032706

Tiempo estimado de no utilización mayor: 0.0127758

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.466713

Reposición LFU - Página menos frecuentemente usada: 0.280028

Reposición LRU - Página menos recientemente usada: 0.233357

Tiempo estimado de no utilización mayor: 0.0131182

Reposición FIFO - Tiempo mayor en memoria principal: 0.00678434

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.405847

Reposición NUR - Página no usada recientemente: 0.279193

Reposición LFU - Página menos frecuentemente usada: 0.167516

Reposición LRU - Página menos recientemente usada: 0.139597

Tiempo estimado de no utilización mayor: 0.00784743

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.411064

Reposición NUR - Página no usada recientemente: 0.282782

Reposición LFU - Página menos frecuentemente usada: 0.169669

Reposición LRU - Página menos recientemente usada: 0.128537

Tiempo estimado de no utilización mayor: 0.0079483

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.421461

Reposición NUR - Página no usada recientemente: 0.289934

Reposición LFU - Página menos frecuentemente usada: 0.17396

Reposición LRU - Página menos recientemente usada: 0.106496

Tiempo estimado de no utilización mayor: 0.00814934

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.781452

Reposición FIFO - Tiempo mayor en memoria principal: 0.103088

Reposición NUR - Página no usada recientemente: 0.0709168

Reposición LFU - Página menos frecuentemente usada: 0.0425501

Tiempo estimado de no utilización mayor: 0.0019933

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.68425

Reposición FIFO - Tiempo mayor en memoria principal: 0.148937

Reposición NUR - Página no usada recientemente: 0.102458

Reposición LFU - Página menos frecuentemente usada: 0.0614748

Tiempo estimado de no utilización mayor: 0.00287984

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.522554

Reposición FIFO - Tiempo mayor en memoria principal: 0.225208

Reposición NUR - Página no usada recientemente: 0.154927

Reposición LFU - Página menos frecuentemente usada: 0.0929561

Tiempo estimado de no utilización mayor: 0.00435462

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.327063

Reposición LRU - Página menos recientemente usada: 0.306621

Reposición NUR - Página no usada recientemente: 0.224995

Reposición LFU - Página menos frecuentemente usada: 0.134997

Tiempo estimado de no utilización mayor: 0.00632406

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.415954

Reposición NUR - Página no usada recientemente: 0.286146

Reposición LFU - Página menos frecuentemente usada: 0.171688

Reposición LRU - Página menos recientemente usada: 0.118169

Tiempo estimado de no utilización mayor: 0.00804287

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.462307

Reposición NUR - Página no usada recientemente: 0.318034

Reposición LFU - Página menos frecuentemente usada: 0.19082

Reposición LRU - Página menos recientemente usada: 0.0198996

Tiempo estimado de no utilización mayor: 0.00893915

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.471694

Reposición NUR - Página no usada recientemente: 0.324491

Reposición LFU - Página menos frecuentemente usada: 0.194695

Tiempo estimado de no utilización mayor: 0.00912064

Reposición LRU - Página menos recientemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.585733

Reposición NUR - Página no usada recientemente: 0.402941

Tiempo estimado de no utilización mayor: 0.0113257

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.585733

Reposición NUR - Página no usada recientemente: 0.402941

Tiempo estimado de no utilización mayor: 0.0113257

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.585733

Reposición NUR - Página no usada recientemente: 0.402941

Tiempo estimado de no utilización mayor: 0.0113257

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.585733

Reposición NUR - Página no usada recientemente: 0.402941

Tiempo estimado de no utilización mayor: 0.0113257

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición FIFO - Tiempo mayor en memoria principal: 0.585733

Reposición NUR - Página no usada recientemente: 0.402941

Tiempo estimado de no utilización mayor: 0.0113257

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Reposición FIFO - Tiempo mayor en memoria principal: 0.981031

Tiempo estimado de no utilización mayor: 0.0189692

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Reposición FIFO - Tiempo mayor en memoria principal: 0.981031

Tiempo estimado de no utilización mayor: 0.0189692

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Reposición FIFO - Tiempo mayor en memoria principal: 0.981031

Tiempo estimado de no utilización mayor: 0.0189692

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Reposición FIFO - Tiempo mayor en memoria principal: 0.981031

Tiempo estimado de no utilización mayor: 0.0189692

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

****

(* Ejemplo de página con:*)

(* 20.001 - 40.000 miliseg. hasta el próximo uso *)

(* 3.001 - 6.000 miliseg. en memoria principal *)

(* -5.000 miliseg. en mem. ppal. sin uso *)

(* 101 - 200 veces de utilización *)

(* sí referenciada y no modificada *)

DatosPagina =

{-1, -2, -3, 4, -5,

-6, -7, 8, -9, -10, -11,

12, -13, -14, -15, -16, -17, -18, -19, -20,

-21, -22, 23, -24, -25,

-26, 27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.654242

Reposición NUR - Página no usada recientemente: 0.138303

Reposición LFU - Página menos frecuentemente usada: 0.0829818

Reposición LRU - Página menos recientemente usada: 0.0691515

Reposición FIFO - Tiempo mayor en memoria principal: 0.0553212

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.365108

Reposición LFU - Página menos frecuentemente usada: 0.219065

Reposición LRU - Página menos recientemente usada: 0.182554

Reposición FIFO - Tiempo mayor en memoria principal: 0.146043

Tiempo estimado de no utilización mayor: 0.0872295

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.370021

Reposición LFU - Página menos frecuentemente usada: 0.222012

Reposición LRU - Página menos recientemente usada: 0.18501

Reposición FIFO - Tiempo mayor en memoria principal: 0.134553

Tiempo estimado de no utilización mayor: 0.0884032

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.379829

Reposición LFU - Página menos frecuentemente usada: 0.227898

Reposición LRU - Página menos recientemente usada: 0.189915

Reposición FIFO - Tiempo mayor en memoria principal: 0.111612

Tiempo estimado de no utilización mayor: 0.0907466

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.834035

Reposición NUR - Página no usada recientemente: 0.0709581

Reposición LFU - Página menos frecuentemente usada: 0.0425749

Reposición LRU - Página menos recientemente usada: 0.0354791

Tiempo estimado de no utilización mayor: 0.0169529

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.670016

Reposición NUR - Página no usada recientemente: 0.141084

Reposición LFU - Página menos frecuentemente usada: 0.0846505

Reposición LRU - Página menos recientemente usada: 0.0705421

Tiempo estimado de no utilización mayor: 0.033707

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.303185

Reposición FIFO - Tiempo mayor en memoria principal: 0.290877

Reposición LFU - Página menos frecuentemente usada: 0.181911

Reposición LRU - Página menos recientemente usada: 0.151592

Tiempo estimado de no utilización mayor: 0.0724351

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.427549

Reposición LFU - Página menos frecuentemente usada: 0.256529

Reposición LRU - Página menos recientemente usada: 0.213774

Tiempo estimado de no utilización mayor: 0.102147

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.73111

Reposición NUR - Página no usada recientemente: 0.146222

Reposición LFU - Página menos frecuentemente usada: 0.0877332

Tiempo estimado de no utilización mayor: 0.0349345

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.687223

Reposición NUR - Página no usada recientemente: 0.170088

Reposición LFU - Página menos frecuentemente usada: 0.102053

Tiempo estimado de no utilización mayor: 0.0406364

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.608389

Reposición NUR - Página no usada recientemente: 0.212958

Reposición LFU - Página menos frecuentemente usada: 0.127775

Tiempo estimado de no utilización mayor: 0.0508786

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.484947

Reposición NUR - Página no usada recientemente: 0.280085

Reposición LFU - Página menos frecuentemente usada: 0.168051

Tiempo estimado de no utilización mayor: 0.0669164

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.368545

Reposición LRU - Página menos recientemente usada: 0.322277

Reposición LFU - Página menos frecuentemente usada: 0.221127

Tiempo estimado de no utilización mayor: 0.0880507

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.456156

Reposición LFU - Página menos frecuentemente usada: 0.273694

Reposición LRU - Página menos recientemente usada: 0.161167

Tiempo estimado de no utilización mayor: 0.108982

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.51388

Reposición LFU - Página menos frecuentemente usada: 0.308328

Tiempo estimado de no utilización mayor: 0.122773

Reposición LRU - Página menos recientemente usada: 0.0550188

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.539044

Reposición LFU - Página menos frecuentemente usada: 0.323426

Tiempo estimado de no utilización mayor: 0.128785

Reposición LRU - Página menos recientemente usada: 0.00874439

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.543799

Reposición LFU - Página menos frecuentemente usada: 0.32628

Tiempo estimado de no utilización mayor: 0.129921

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

(* Ejemplo de página con:*)

(* 20.001 - 40.000 miliseg. hasta el próximo uso *)

(* 3.001 - 6.000 miliseg. en memoria principal *)

(* -5.000 miliseg. en mem. ppal. sin uso *)

(* 101 - 200 veces de utilización *)

(* no referenciada *)

DatosPagina =

{-1, -2, -3, 4, -5,

-6, -7, 8, -9, -10, -11,

12, -13, -14, -15, -16, -17, -18, -19, -20,

-21, -22, 23, -24, -25,

26, -27, -28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.654242

Reposición NUR - Página no usada recientemente: 0.138303

Reposición LFU - Página menos frecuentemente usada: 0.0829818

Reposición LRU - Página menos recientemente usada: 0.0691515

Reposición FIFO - Tiempo mayor en memoria principal: 0.0553212

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.365108

Reposición LFU - Página menos frecuentemente usada: 0.219065

Reposición LRU - Página menos recientemente usada: 0.182554

Reposición FIFO - Tiempo mayor en memoria principal: 0.146043

Tiempo estimado de no utilización mayor: 0.0872295

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.370021

Reposición LFU - Página menos frecuentemente usada: 0.222012

Reposición LRU - Página menos recientemente usada: 0.18501

Reposición FIFO - Tiempo mayor en memoria principal: 0.134553

Tiempo estimado de no utilización mayor: 0.0884032

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.379829

Reposición LFU - Página menos frecuentemente usada: 0.227898

Reposición LRU - Página menos recientemente usada: 0.189915

Reposición FIFO - Tiempo mayor en memoria principal: 0.111612

Tiempo estimado de no utilización mayor: 0.0907466

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.834035

Reposición NUR - Página no usada recientemente: 0.0709581

Reposición LFU - Página menos frecuentemente usada: 0.0425749

Reposición LRU - Página menos recientemente usada: 0.0354791

Tiempo estimado de no utilización mayor: 0.0169529

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.670016

Reposición NUR - Página no usada recientemente: 0.141084

Reposición LFU - Página menos frecuentemente usada: 0.0846505

Reposición LRU - Página menos recientemente usada: 0.0705421

Tiempo estimado de no utilización mayor: 0.033707

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.303185

Reposición FIFO - Tiempo mayor en memoria principal: 0.290877

Reposición LFU - Página menos frecuentemente usada: 0.181911

Reposición LRU - Página menos recientemente usada: 0.151592

Tiempo estimado de no utilización mayor: 0.0724351

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.427549

Reposición LFU - Página menos frecuentemente usada: 0.256529

Reposición LRU - Página menos recientemente usada: 0.213774

Tiempo estimado de no utilización mayor: 0.102147

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.73111

Reposición NUR - Página no usada recientemente: 0.146222

Reposición LFU - Página menos frecuentemente usada: 0.0877332

Tiempo estimado de no utilización mayor: 0.0349345

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.687223

Reposición NUR - Página no usada recientemente: 0.170088

Reposición LFU - Página menos frecuentemente usada: 0.102053

Tiempo estimado de no utilización mayor: 0.0406364

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.608389

Reposición NUR - Página no usada recientemente: 0.212958

Reposición LFU - Página menos frecuentemente usada: 0.127775

Tiempo estimado de no utilización mayor: 0.0508786

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.484947

Reposición NUR - Página no usada recientemente: 0.280085

Reposición LFU - Página menos frecuentemente usada: 0.168051

Tiempo estimado de no utilización mayor: 0.0669164

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.368545

Reposición LRU - Página menos recientemente usada: 0.322277

Reposición LFU - Página menos frecuentemente usada: 0.221127

Tiempo estimado de no utilización mayor: 0.0880507

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.456156

Reposición LFU - Página menos frecuentemente usada: 0.273694

Reposición LRU - Página menos recientemente usada: 0.161167

Tiempo estimado de no utilización mayor: 0.108982

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.51388

Reposición LFU - Página menos frecuentemente usada: 0.308328

Tiempo estimado de no utilización mayor: 0.122773

Reposición LRU - Página menos recientemente usada: 0.0550188

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.539044

Reposición LFU - Página menos frecuentemente usada: 0.323426

Tiempo estimado de no utilización mayor: 0.128785

Reposición LRU - Página menos recientemente usada: 0.00874439

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.543799

Reposición LFU - Página menos frecuentemente usada: 0.32628

Tiempo estimado de no utilización mayor: 0.129921

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Reposición NUR - Página no usada recientemente: 0.997617

Tiempo estimado de no utilización mayor: 0.00238345

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Reposición NUR - Página no usada recientemente: 0.995292

Tiempo estimado de no utilización mayor: 0.00470823

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Reposición NUR - Página no usada recientemente: 0.995095

Tiempo estimado de no utilización mayor: 0.0049055

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Reposición NUR - Página no usada recientemente: 0.995095

Tiempo estimado de no utilización mayor: 0.0049055

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

(* Ejemplo de página con:*)

(* 20.001 - 40.000 miliseg. hasta el próximo uso *)

(* 3.001 - 6.000 miliseg. en memoria principal *)

(* -5.000 miliseg. en mem. ppal. sin uso *)

(* 101 - 200 veces de utilización *)

(* sí referenciada y sí modificada *)

DatosPagina =

{-1, -2, -3, 4, -5,

-6, -7, 8, -9, -10, -11,

12, -13, -14, -15, -16, -17, -18, -19, -20,

-21, -22, 23, -24, -25,

-26, -27, 28};

Paginacion[Prior, Verosimilitudes, DatosPagina]

****

* Análisis del encuadre de la página *

* según las estrategias de paginación, *

* lo que determinará la posibilidad de ser *

* desalojada de la memoria principal *

* según las distintas estrategias. *

****

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: - 5.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.374291

Reposición LFU - Página menos frecuentemente usada: 0.224575

Reposición LRU - Página menos recientemente usada: 0.187146

Reposición FIFO - Tiempo mayor en memoria principal: 0.149716

Tiempo estimado de no utilización mayor: 0.0642722

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 5.000 - 10.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.380215

Reposición LFU - Página menos frecuentemente usada: 0.228129

Reposición LRU - Página menos recientemente usada: 0.190108

Reposición FIFO - Tiempo mayor en memoria principal: 0.152086

Tiempo estimado de no utilización mayor: 0.0494617

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 10.001 - 20.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.387771

Reposición LFU - Página menos frecuentemente usada: 0.232663

Reposición LRU - Página menos recientemente usada: 0.193885

Reposición FIFO - Tiempo mayor en memoria principal: 0.155108

Tiempo estimado de no utilización mayor: 0.0305725

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: 20.001 - 40.000 miliseg.

Tiempo estimado de no utilización mayor: 0.654242

Reposición NUR - Página no usada recientemente: 0.138303

Reposición LFU - Página menos frecuentemente usada: 0.0829818

Reposición LRU - Página menos recientemente usada: 0.0691515

Reposición FIFO - Tiempo mayor en memoria principal: 0.0553212

Resultados tras la consideración de la pauta:

Tiempo hasta la próxima utilización: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.365108

Reposición LFU - Página menos frecuentemente usada: 0.219065

Reposición LRU - Página menos recientemente usada: 0.182554

Reposición FIFO - Tiempo mayor en memoria principal: 0.146043

Tiempo estimado de no utilización mayor: 0.0872295

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: - 1.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.370021

Reposición LFU - Página menos frecuentemente usada: 0.222012

Reposición LRU - Página menos recientemente usada: 0.18501

Reposición FIFO - Tiempo mayor en memoria principal: 0.134553

Tiempo estimado de no utilización mayor: 0.0884032

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 1.000 - 3.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.379829

Reposición LFU - Página menos frecuentemente usada: 0.227898

Reposición LRU - Página menos recientemente usada: 0.189915

Reposición FIFO - Tiempo mayor en memoria principal: 0.111612

Tiempo estimado de no utilización mayor: 0.0907466

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 3.001 - 6.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.834035

Reposición NUR - Página no usada recientemente: 0.0709581

Reposición LFU - Página menos frecuentemente usada: 0.0425749

Reposición LRU - Página menos recientemente usada: 0.0354791

Tiempo estimado de no utilización mayor: 0.0169529

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 6.001 - 9.000 miliseg.

Reposición FIFO - Tiempo mayor en memoria principal: 0.670016

Reposición NUR - Página no usada recientemente: 0.141084

Reposición LFU - Página menos frecuentemente usada: 0.0846505

Reposición LRU - Página menos recientemente usada: 0.0705421

Tiempo estimado de no utilización mayor: 0.033707

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: 9.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.303185

Reposición FIFO - Tiempo mayor en memoria principal: 0.290877

Reposición LFU - Página menos frecuentemente usada: 0.181911

Reposición LRU - Página menos recientemente usada: 0.151592

Tiempo estimado de no utilización mayor: 0.0724351

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal.: + 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.427549

Reposición LFU - Página menos frecuentemente usada: 0.256529

Reposición LRU - Página menos recientemente usada: 0.213774

Tiempo estimado de no utilización mayor: 0.102147

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: - 5.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.73111

Reposición NUR - Página no usada recientemente: 0.146222

Reposición LFU - Página menos frecuentemente usada: 0.0877332

Tiempo estimado de no utilización mayor: 0.0349345

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 5.000 - 10.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.687223

Reposición NUR - Página no usada recientemente: 0.170088

Reposición LFU - Página menos frecuentemente usada: 0.102053

Tiempo estimado de no utilización mayor: 0.0406364

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 10.001 - 15.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.608389

Reposición NUR - Página no usada recientemente: 0.212958

Reposición LFU - Página menos frecuentemente usada: 0.127775

Tiempo estimado de no utilización mayor: 0.0508786

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 15.001 - 20.000 miliseg.

Reposición LRU - Página menos recientemente usada: 0.484947

Reposición NUR - Página no usada recientemente: 0.280085

Reposición LFU - Página menos frecuentemente usada: 0.168051

Tiempo estimado de no utilización mayor: 0.0669164

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 20.001 - 25.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.368545

Reposición LRU - Página menos recientemente usada: 0.322277

Reposición LFU - Página menos frecuentemente usada: 0.221127

Tiempo estimado de no utilización mayor: 0.0880507

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 25.001 - 30.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.456156

Reposición LFU - Página menos frecuentemente usada: 0.273694

Reposición LRU - Página menos recientemente usada: 0.161167

Tiempo estimado de no utilización mayor: 0.108982

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 30.001 - 35.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.51388

Reposición LFU - Página menos frecuentemente usada: 0.308328

Tiempo estimado de no utilización mayor: 0.122773

Reposición LRU - Página menos recientemente usada: 0.0550188

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: 35.001 - 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.539044

Reposición LFU - Página menos frecuentemente usada: 0.323426

Tiempo estimado de no utilización mayor: 0.128785

Reposición LRU - Página menos recientemente usada: 0.00874439

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Tiempo en mem. ppal. s/ uso: + 40.000 miliseg.

Reposición NUR - Página no usada recientemente: 0.543799

Reposición LFU - Página menos frecuentemente usada: 0.32628

Tiempo estimado de no utilización mayor: 0.129921

Reposición LRU - Página menos recientemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Menos de 50 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 50 y 100 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 101 y 200 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Entre 201 y 300 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Intensidad de utilización de la página: Más de 300 veces

Reposición NUR - Página no usada recientemente: 0.807158

Tiempo estimado de no utilización mayor: 0.192842

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página no referenciada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y no modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

Resultados tras la consideración de la pauta:

Página sí referenciada y sí modificada

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Resultado final:

La página tiene las siguientes probabilidades

de ser removida de la memoria principal según

las estrategias consideradas:

Tiempo estimado de no utilización mayor: 1.

Reposición NUR - Página no usada recientemente: 0.

Reposición LRU - Página menos recientemente usada: 0.

Reposición LFU - Página menos frecuentemente usada: 0.

Reposición FIFO - Tiempo mayor en memoria principal: 0.

****

Capítulo 24

Análisis del Rendimiento de un Subsistema de Disco de Una Petición a la Vez con Mathematica

24.1 Datos y Ejecuciones

Contenido del archivo **Colas1en.ma:**

<<Examples‘Colas1pn‘

? Colas1pn

Colas1pn[mu,lambda,i]

Análisis del rendimiento de un subsistema de disco que solo puede dar servicio a una petición a la vez.

Colocar los valores máximos para mu, lambda y el número de peticiones pendientes.

Colas1pn[20,10,3]

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009}}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7, 0.21, 0.063, 0.0189}}

El promedio de peticiones pendientes es el siguiente:

0.428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.5, 0.25, 0.125, 0.0625}}

El promedio de peticiones pendientes es el siguiente:

1.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.3, 0.21, 0.147, 0.1029}}

El promedio de peticiones pendientes es el siguiente:

2.33333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.1, 0.09, 0.081, 0.0729}}

El promedio de peticiones pendientes es el siguiente:

9.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.95, 0.0475, 0.002375, 0.00011875}}

El promedio de peticiones pendientes es el siguiente:

0.0526316

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.85, 0.1275, 0.019125, 0.00286875}}

El promedio de peticiones pendientes es el siguiente:

0.176471

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.75, 0.1875, 0.046875, 0.0117188}}

El promedio de peticiones pendientes es el siguiente:

0.333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.65, 0.2275, 0.079625, 0.0278688}}

El promedio de peticiones pendientes es el siguiente:

0.538462

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.55, 0.2475, 0.111375, 0.0501187}}

El promedio de peticiones pendientes es el siguiente:

0.818182

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45}

Promedio de peticiones pendientes para cada coeficiente:

{0.111111, 0.428571, 1., 2.33333, 9., 0.0526316, 0.176471, 0.333333, 0.538462, 0.818182}

Pares (coeficientes,promedio):

{{0.1, 0.111111}, {0.3, 0.428571}, {0.5, 1.}, {0.7, 2.33333}, {0.9, 9.}, {0.05, 0.0526316}, {0.15, 0.176471}, {0.25, 0.333333}, {0.35, 0.538462}, {0.45, 0.818182}}

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

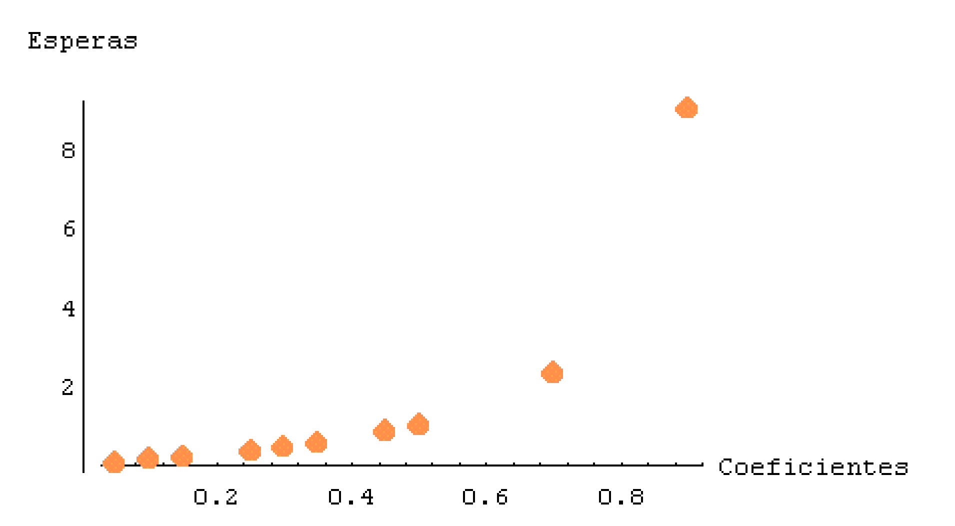


Figura 24.1: Promedio de peticiones pendientes.

{Statistics‘LinearRegression‘BestFit ->

-0.706132 + 12.4786 x - 41.07 x^2 + 43.3673 x^3 ,

Statistics‘LinearRegression‘BestFitCoefficients ->

{-0.706132, 12.4786, -41.07, 43.3673}}

Colas1pn[40,20,7]

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009, 0.00009, $9.\ 10^{-6}$, $9.\ 10^{-7}$, $9.\ 10^{-8}$ }}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7, 0.21, 0.063, 0.0189, 0.00567, 0.001701, 0.0005103, 0.00015309}}

El promedio de peticiones pendientes es el siguiente:

0.428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125, 0.00390625}}

El promedio de peticiones pendientes es el siguiente:

1.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.3, 0.21, 0.147, 0.1029, 0.07203, 0.050421, 0.0352947, 0.0247063}}

El promedio de peticiones pendientes es el siguiente:

2.33333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.1, 0.09, 0.081, 0.0729, 0.06561, 0.059049, 0.0531441, 0.0478297}}

El promedio de peticiones pendientes es el siguiente:

9.

****

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.95, 0.0475, 0.002375, 0.00011875, 5.9375 10^{-6} , 2.96875 10^{-7} , 1.48437 10^{-8} , 7.42187 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.0526316

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.85, 0.1275, 0.019125, 0.00286875, 0.000430312, 0.0000645469, 9.68203 10^{-6} , 1.4523 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.176471

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.75, 0.1875, 0.046875, 0.0117188, 0.00292969, 0.000732422, 0.000183105, 0.0000457764}}

El promedio de peticiones pendientes es el siguiente:

0.333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.65, 0.2275, 0.079625, 0.0278688, 0.00975406, 0.00341392, 0.00119487, 0.000418205}}

El promedio de peticiones pendientes es el siguiente:

0.538462

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.55, 0.2475, 0.111375, 0.0501187, 0.0225534, 0.010149, 0.00456707, 0.00205518}}

El promedio de peticiones pendientes es el siguiente:

0.818182

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 11., 0.55}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.45, 0.2475, 0.136125, 0.0748687, 0.0411778, 0.0226478, 0.0124563, 0.00685096}}

El promedio de peticiones pendientes es el siguiente:

1.22222

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 13., 0.65}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.35, 0.2275, 0.147875, 0.0961187, 0.0624772, 0.0406102, 0.0263966, 0.0171578}}

El promedio de peticiones pendientes es el siguiente:

1.85714

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 15., 0.75}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.25, 0.1875, 0.140625, 0.105469, 0.0791016, 0.0593262, 0.0444946, 0.033371}}

El promedio de peticiones pendientes es el siguiente:

3.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 17., 0.85}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.15, 0.1275, 0.108375, 0.0921188, 0.0783009, 0.0665558, 0.0565724, 0.0480866}}

El promedio de peticiones pendientes es el siguiente:

5.66667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 19., 0.95}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.05, 0.0475, 0.045125, 0.0428688, 0.0407253, 0.038689, 0.0367546, 0.0349169}}

El promedio de peticiones pendientes es el siguiente:

19.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 1., 0.0333333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.966667, 0.0322222, 0.00107407, 0.0000358025, 1.19342 10^{-6} , 3.97805 10^{-8} , 1.32602 10^{-9} , 4.42006 10^{-11} }}

El promedio de peticiones pendientes es el siguiente:

0.0344828

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 3., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009, 0.00009, 9. 10^{-6} , 9. 10^{-7} , 9. 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 5., 0.166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.833333, 0.138889, 0.0231481, 0.00385802, 0.000643004, 0.000107167, 0.0000178612, 2.97687 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.2

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 7., 0.233333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.766667, 0.178889, 0.0417407, 0.00973951, 0.00227255, 0.000530262, 0.000123728, 0.0000288698}}

El promedio de peticiones pendientes es el siguiente:

0.304348

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 9., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7, 0.21, 0.063, 0.0189, 0.00567, 0.001701, 0.0005103, 0.00015309}}

El promedio de peticiones pendientes es el siguiente:

0.428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 11., 0.366667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.633333, 0.232222, 0.0851481, 0.031221, 0.0114477, 0.00419749, 0.00153908, 0.000564329}}

El promedio de peticiones pendientes es el siguiente:

0.578947

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 13., 0.433333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.566667, 0.245556, 0.106407, 0.0461099, 0.0199809, 0.00865841, 0.00375198, 0.00162586}}

El promedio de peticiones pendientes es el siguiente:

0.764706

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 15., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125, 0.00390625}}

El promedio de peticiones pendientes es el siguiente:

1.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 17., 0.566667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.433333, 0.245556, 0.139148, 0.0788506, 0.044682, 0.0253198, 0.0143479, 0.00813047}}

El promedio de peticiones pendientes es el siguiente:

1.30769

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 19., 0.633333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.366667, 0.232222, 0.147074, 0.0931469, 0.058993, 0.0373623, 0.0236628, 0.0149864}}

El promedio de peticiones pendientes es el siguiente:

1.72727

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 1., 0.025}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.975, 0.024375, 0.000609375, 0.0000152344, 3.80859 10^{-7} , 9.52148 10^{-9} , 2.38037 10^{-10} , 5.95093 10^{-12} }}

El promedio de peticiones pendientes es el siguiente:

0.025641

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 3., 0.075}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.925, 0.069375, 0.00520313, 0.000390234, 0.0000292676, 2.19507 10^{-6} , 1.6463 10^{-7} , 1.23473 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.0810811

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 5., 0.125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.875, 0.109375, 0.0136719, 0.00170898, 0.000213623, 0.0000267029, 3.33786 10^{-6} , 4.17233 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.142857

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 7., 0.175}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.825, 0.144375, 0.0252656, 0.00442148, 0.00077376, 0.000135408, 0.0000236964, 4.14687 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.212121

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 9., 0.225}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.775, 0.174375, 0.0392344, 0.00882773, 0.00198624, 0.000446904, 0.000100553,

0.0000226245}}

El promedio de peticiones pendientes es el siguiente:

0.290323

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 11., 0.275}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.725, 0.199375, 0.0548281, 0.0150777, 0.00414638, 0.00114025, 0.00031357, 0.0000862317}}

El promedio de peticiones pendientes es el siguiente:

0.37931

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 13., 0.325}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.675, 0.219375, 0.0712969, 0.0231715, 0.00753073, 0.00244749, 0.000795434, 0.000258516}}

El promedio de peticiones pendientes es el siguiente:

0.481481

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 15., 0.375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.625, 0.234375, 0.0878906, 0.032959, 0.0123596, 0.00463486, 0.00173807, 0.000651777}}

El promedio de peticiones pendientes es el siguiente:

0.6

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 17., 0.425}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.575, 0.244375, 0.103859, 0.0441402, 0.0187596, 0.00797283, 0.00338845, 0.00144009}}

El promedio de peticiones pendientes es el siguiente:

0.73913

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 19., 0.475}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.525, 0.249375, 0.118453, 0.0562652, 0.026726, 0.0126948, 0.00603005, 0.00286427}}

El promedio de peticiones pendientes es el siguiente:

0.904762

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 0.0333333, 0.1, 0.166667, 0.233333, 0.3, 0.366667, 0.433333, 0.5, 0.566667, 0.633333, 0.025, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, 0.425, 0.475}

Promedio de peticiones pendientes para cada coeficiente:

{0.111111, 0.428571, 1., 2.33333, 9., 0.0526316, 0.176471, 0.333333, 0.538462, 0.818182, 1.22222, 1.85714, 3., 5.66667, 19., 0.0344828, 0.111111, 0.2, 0.304348, 0.428571, 0.578947, 0.764706, 1., 1.30769, 1.72727, 0.025641, 0.0810811, 0.142857, 0.212121, 0.290323, 0.37931, 0.481481, 0.6, 0.73913, 0.904762}

Pares (coeficientes,promedio):

{{0.1, 0.111111}, {0.3, 0.428571}, {0.5, 1.}, {0.7, 2.33333}, {0.9, 9.}, {0.05, 0.0526316},

{0.15, 0.176471}, {0.25, 0.333333}, {0.35, 0.538462}, {0.45, 0.818182},

{0.55, 1.22222}, {0.65, 1.85714}, {0.75, 3.}, {0.85, 5.66667}, {0.95, 19.},

{0.0333333, 0.0344828}, {0.1, 0.111111}, {0.166667, 0.2}, {0.233333, 0.304348},

{0.3, 0.428571}, {0.366667, 0.578947}, {0.433333, 0.764706}, {0.5, 1.},

{0.566667, 1.30769}, {0.633333, 1.72727}, {0.025, 0.025641}, {0.075, 0.0810811},

{0.125, 0.142857}, {0.175, 0.212121}, {0.225, 0.290323}, {0.275, 0.37931},

{0.325, 0.481481}, {0.375, 0.6}, {0.425, 0.73913}, {0.475, 0.904762}}

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

{Statistics‘LinearRegression‘BestFit ->

-1.34974 + 23.4148 x - 77.7456 x^2 + 75.0338 x^3 ,

Statistics‘LinearRegression‘BestFitCoefficien ts ->

{-1.34974, 23.4148, -77.7456, 75.0338}}

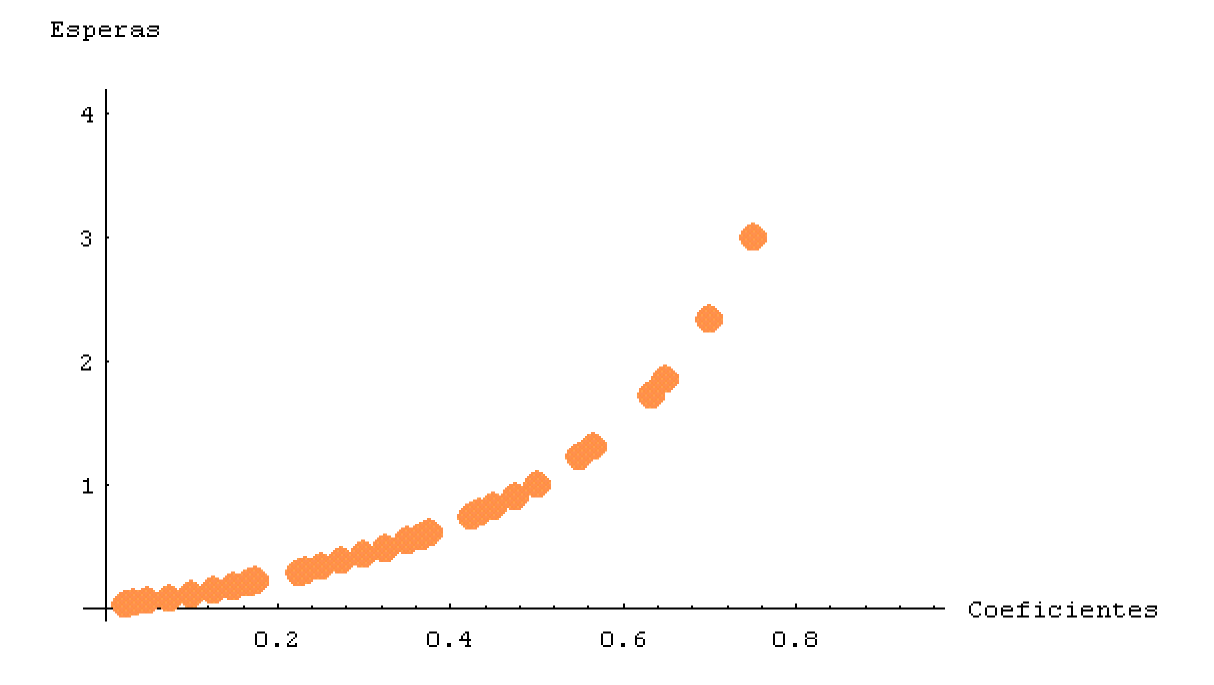


Figura 24.2: Promedio de peticiones pendientes.

Colas1pn[90,20,5]

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009, 0.00009, 9. 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7, 0.21, 0.063, 0.0189, 0.00567, 0.001701}}

El promedio de peticiones pendientes es el siguiente:

0.428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625}}

El promedio de peticiones pendientes es el siguiente:

1.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.3, 0.21, 0.147, 0.1029, 0.07203, 0.050421}}

El promedio de peticiones pendientes es el siguiente:

2.33333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.1, 0.09, 0.081, 0.0729, 0.06561, 0.059049}}

El promedio de peticiones pendientes es el siguiente:

9.

****

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.95, 0.0475, 0.002375, 0.00011875, 5.9375 10^{-6} , 2.96875 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.0526316

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.85, 0.1275, 0.019125, 0.00286875, 0.000430312, 0.0000645469}}

El promedio de peticiones pendientes es el siguiente:

0.176471

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.75, 0.1875, 0.046875, 0.0117188, 0.00292969, 0.000732422}}

El promedio de peticiones pendientes es el siguiente:

0.333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.65, 0.2275, 0.079625, 0.0278688, 0.00975406, 0.00341392}}

El promedio de peticiones pendientes es el siguiente:

0.538462

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.55, 0.2475, 0.111375, 0.0501187, 0.0225534, 0.010149}}

El promedio de peticiones pendientes es el siguiente:

0.818182

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 11., 0.55}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.45, 0.2475, 0.136125, 0.0748687, 0.0411778, 0.0226478}}

El promedio de peticiones pendientes es el siguiente:

1.22222

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 13., 0.65}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.35, 0.2275, 0.147875, 0.0961187, 0.0624772, 0.0406102}}

El promedio de peticiones pendientes es el siguiente:

1.85714

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 15., 0.75}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.25, 0.1875, 0.140625, 0.105469, 0.0791016, 0.0593262}}

El promedio de peticiones pendientes es el siguiente:

3.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 17., 0.85}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.15, 0.1275, 0.108375, 0.0921188, 0.0783009, 0.0665558}}

El promedio de peticiones pendientes es el siguiente:

5.66667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 19., 0.95}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.05, 0.0475, 0.045125, 0.0428688, 0.0407253, 0.038689}}

El promedio de peticiones pendientes es el siguiente:

19.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 1., 0.0333333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.966667, 0.0322222, 0.00107407, 0.0000358025, 1.19342 10^{-6} , 3.97805 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.0344828

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 3., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009, 0.00009, $9.\ 10^{-6}$ }}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 5., 0.166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.833333, 0.138889, 0.0231481, 0.00385802, 0.000643004, 0.000107167}}

El promedio de peticiones pendientes es el siguiente:

0.2

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 7., 0.233333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.766667, 0.178889, 0.0417407, 0.00973951, 0.00227255, 0.000530262}}

El promedio de peticiones pendientes es el siguiente:

0.304348

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 9., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7, 0.21, 0.063, 0.0189, 0.00567, 0.001701}}

El promedio de peticiones pendientes es el siguiente:

0.428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 11., 0.366667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.633333, 0.232222, 0.0851481, 0.031221, 0.0114477, 0.00419749}}

El promedio de peticiones pendientes es el siguiente:

0.578947

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 13., 0.433333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.566667, 0.245556, 0.106407, 0.0461099, 0.0199809, 0.00865841}}

El promedio de peticiones pendientes es el siguiente:

0.764706

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 15., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625}}

El promedio de peticiones pendientes es el siguiente:

1.

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 17., 0.566667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.433333, 0.245556, 0.139148, 0.0788506, 0.044682, 0.0253198}}

El promedio de peticiones pendientes es el siguiente:

1.30769

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 19., 0.633333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.366667, 0.232222, 0.147074, 0.0931469, 0.058993, 0.0373623}}

El promedio de peticiones pendientes es el siguiente:

1.72727

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 1., 0.025}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.975, 0.024375, 0.000609375, 0.0000152344, 3.80859 10^{-7} , 9.52148 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.025641

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 3., 0.075}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.925, 0.069375, 0.00520313, 0.000390234, 0.0000292676, 2.19507 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.0810811

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 5., 0.125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.875, 0.109375, 0.0136719, 0.00170898, 0.000213623, 0.0000267029}}

El promedio de peticiones pendientes es el siguiente:

0.142857

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 7., 0.175}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.825, 0.144375, 0.0252656, 0.00442148, 0.00077376, 0.000135408}}

El promedio de peticiones pendientes es el siguiente:

0.212121

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 9., 0.225}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.775, 0.174375, 0.0392344, 0.00882773, 0.00198624, 0.000446904}}

El promedio de peticiones pendientes es el siguiente:

0.290323

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 11., 0.275}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.725, 0.199375, 0.0548281, 0.0150777, 0.00414638, 0.00114025}}

El promedio de peticiones pendientes es el siguiente:

0.37931

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 13., 0.325}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.675, 0.219375, 0.0712969, 0.0231715, 0.00753073, 0.00244749}}

El promedio de peticiones pendientes es el siguiente:

0.481481

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 15., 0.375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.625, 0.234375, 0.0878906, 0.032959, 0.0123596, 0.00463486}}

El promedio de peticiones pendientes es el siguiente:

0.6

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 17., 0.425}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.575, 0.244375, 0.103859, 0.0441402, 0.0187596, 0.00797283}}

El promedio de peticiones pendientes es el siguiente:

0.73913

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 19., 0.475}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.525, 0.249375, 0.118453, 0.0562652, 0.026726, 0.0126948}}

El promedio de peticiones pendientes es el siguiente:

0.904762

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 1., 0.02}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.98, 0.0196, 0.000392, 7.84 10^{-6} , 1.568 10^{-7} , 3.136 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.0204082

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 3., 0.06}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.94, 0.0564, 0.003384, 0.00020304, 0.0000121824, 7.30944 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.0638298

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 5., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009, 0.00009, 9. 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 7., 0.14}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.86, 0.1204, 0.016856, 0.00235984, 0.000330378, 0.0000462529}}

El promedio de peticiones pendientes es el siguiente:

0.162791

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 9., 0.18}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.82, 0.1476, 0.026568, 0.00478224, 0.000860803, 0.000154945}}

El promedio de peticiones pendientes es el siguiente:

0.219512

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 11., 0.22}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.78, 0.1716, 0.037752, 0.00830544, 0.0018272, 0.000401983}}

El promedio de peticiones pendientes es el siguiente:

0.282051

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 13., 0.26}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.74, 0.1924, 0.050024, 0.0130062, 0.00338162, 0.000879222}}

El promedio de peticiones pendientes es el siguiente:

0.351351

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 15., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7, 0.21, 0.063, 0.0189, 0.00567, 0.001701}}

El promedio de peticiones pendientes es el siguiente:

0.428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 17., 0.34}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.66, 0.2244, 0.076296, 0.0259406, 0.00881982, 0.00299874}}

El promedio de peticiones pendientes es el siguiente:

0.515152

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 19., 0.38}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.62, 0.2356, 0.089528, 0.0340206, 0.0129278, 0.00491258}}

El promedio de peticiones pendientes es el siguiente:

0.612903

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 1., 0.0166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.983333, 0.0163889, 0.000273148, 4.55247 10^{-6} , 7.58745 10^{-8} , 1.26457 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.0169492

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 3., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.95, 0.0475, 0.002375, 0.00011875, 5.9375 10^{-6} , 2.96875 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.0526316

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 5., 0.0833333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.916667, 0.0763889, 0.00636574, 0.000530478, 0.0000442065, 3.68388 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.0909091

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 7., 0.116667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.883333, 0.103056, 0.0120231, 0.0014027, 0.000163648, 0.0000190923}}

El promedio de peticiones pendientes es el siguiente:

0.132075

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 9., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.85, 0.1275, 0.019125, 0.00286875, 0.000430312, 0.0000645469}}

El promedio de peticiones pendientes es el siguiente:

0.176471

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 11., 0.183333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.816667, 0.149722, 0.0274491, 0.00503233, 0.000922594, 0.000169142}}

El promedio de peticiones pendientes es el siguiente:

0.22449

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 13., 0.216667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.783333, 0.169722, 0.0367731, 0.00796752, 0.0017263, 0.000374031}}

El promedio de peticiones pendientes es el siguiente:

0.276596

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 15., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.75, 0.1875, 0.046875, 0.0117188, 0.00292969, 0.000732422}}

El promedio de peticiones pendientes es el siguiente:

0.333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 17., 0.283333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.716667, 0.203056, 0.0575324, 0.0163008, 0.00461857, 0.0013086}}

El promedio de peticiones pendientes es el siguiente:

0.395349

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 19., 0.316667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.683333, 0.216389, 0.0685231, 0.021699, 0.00687135, 0.00217593}}

El promedio de peticiones pendientes es el siguiente:

0.463415

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 1., 0.0142857}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.985714, 0.0140816, 0.000201166, 2.8738 10^{-6} , 4.10543 10^{-8} , 5.8649 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.0144928

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 3., 0.0428571}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.957143, 0.0410204, 0.00175802, 0.0000753436, 3.22901 10^{-6} , 1.38386 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.0447761

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 5., 0.0714286}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.928571, 0.0663265, 0.00473761, 0.000338401, 0.0000241715, 1.72653 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.0769231

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 7., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009, 0.00009, 9. 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 9., 0.128571}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.871429, 0.112041, 0.0144052, 0.0018521, 0.000238128, 0.0000306164}}

El promedio de peticiones pendientes es el siguiente:

0.147541

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 11., 0.157143}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.842857, 0.132449, 0.0208134, 0.00327068, 0.000513964, 0.0000807657}}

El promedio de peticiones pendientes es el siguiente:

0.186441

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 13., 0.185714}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.814286, 0.151224, 0.0280845, 0.0052157, 0.00096863, 0.000179888}}

El promedio de peticiones pendientes es el siguiente:

0.22807

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 15., 0.214286}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.785714, 0.168367, 0.0360787, 0.00773115, 0.00165668, 0.000355002}}

El promedio de peticiones pendientes es el siguiente:

0.272727

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 17., 0.242857}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.757143, 0.183878, 0.044656, 0.010845, 0.00263379, 0.000639635}}

El promedio de peticiones pendientes es el siguiente:

0.320755

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 19., 0.271429}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.728571, 0.197755, 0.0536764, 0.0145693, 0.00395453, 0.00107337}}

El promedio de peticiones pendientes es el siguiente:

0.372549

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 1., 0.0125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9875, 0.0123438, 0.000154297, 1.92871 10^{-6} , 2.41089 10^{-8} , 3.01361 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.0126582

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 3., 0.0375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9625, 0.0360938, 0.00135352, 0.0000507568, 1.90338 10^{-6} , 7.13768 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.038961

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 5., 0.0625}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9375, 0.0585937, 0.00366211, 0.000228882, 0.0000143051, $8.9407\ 10^{-7}$ }}

El promedio de peticiones pendientes es el siguiente:

0.0666667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 7., 0.0875}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9125, 0.0798437, 0.00698633, 0.000611304, 0.0000534891, $4.68029\ 10^{-6}$ }}

El promedio de peticiones pendientes es el siguiente:

0.0958904

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 9., 0.1125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.8875, 0.0998437, 0.0112324, 0.00126365, 0.00014216, 0.000015993}}

El promedio de peticiones pendientes es el siguiente:

0.126761

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 11., 0.1375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.8625, 0.118594, 0.0163066, 0.00224216, 0.000308297, 0.0000423909}}

El promedio de peticiones pendientes es el siguiente:

0.15942

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 13., 0.1625}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.8375, 0.136094, 0.0221152, 0.00359373, 0.00058398, 0.0000948968}}

El promedio de peticiones pendientes es el siguiente:

0.19403

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 15., 0.1875}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.8125, 0.152344, 0.0285645, 0.00535583, 0.00100422, 0.000188291}}

El promedio de peticiones pendientes es el siguiente:

0.230769

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 17., 0.2125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7875, 0.167344, 0.0355605, 0.00755662, 0.00160578, 0.000341228}}

El promedio de peticiones pendientes es el siguiente:

0.269841

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 19., 0.2375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.7625, 0.181094, 0.0430098, 0.0102148, 0.00242602, 0.00057618}}

El promedio de peticiones pendientes es el siguiente:

0.311475

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 1., 0.0111111}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.988889, 0.0109877, 0.000122085, 1.3565 10^{-6} , 1.50722 10^{-8} , 1.67469 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.011236

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 3., 0.0333333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.966667, 0.0322222, 0.00107407, 0.0000358025, 1.19342 10^{-6} , 3.97805 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.0344828

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 5., 0.0555556}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.944444, 0.0524691, 0.00291495, 0.000161942, 8.99677 10^{-6} , 4.9982 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.0588235

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 7., 0.0777778}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.922222, 0.0717284, 0.00557888, 0.000433913, 0.0000337488, 2.6249 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.0843373

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 9., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.9, 0.09, 0.009, 0.0009, 0.00009, $9.\ 10^{-6}$ }}

El promedio de peticiones pendientes es el siguiente:

0.111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 11., 0.122222}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.877778, 0.107284, 0.0131125, 0.00160264, 0.000195878, 0.0000239406}}

El promedio de peticiones pendientes es el siguiente:

0.139241

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 13., 0.144444}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.855556, 0.12358, 0.0178505, 0.0025784, 0.000372436, 0.0000537963}}

El promedio de peticiones pendientes es el siguiente:

0.168831

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 15., 0.166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.833333, 0.138889, 0.0231481, 0.00385802, 0.000643004, 0.000107167}}

El promedio de peticiones pendientes es el siguiente:

0.2

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 17., 0.188889}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.811111, 0.15321, 0.0289396, 0.00546638, 0.00103254, 0.000195035}}

El promedio de peticiones pendientes es el siguiente:

0.232877

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 1:

El subsistema de disco solo puede dar servicio a una petición a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 19., 0.211111}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.788889, 0.166543, 0.0351591, 0.00742248, 0.00156697, 0.000330804}}

El promedio de peticiones pendientes es el siguiente:

0.267606

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 0.0333333, 0.1, 0.166667, 0.233333, 0.3, 0.366667, 0.433333, 0.5, 0.566667, 0.633333, 0.025, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, 0.425, 0.475, 0.02, 0.06, 0.1, 0.14, 0.18, 0.22, 0.26, 0.3, 0.34, 0.38, 0.0166667, 0.05, 0.0833333, 0.116667, 0.15, 0.183333, 0.216667, 0.25, 0.283333, 0.316667, 0.0142857, 0.0428571, 0.0714286, 0.1, 0.128571, 0.157143, 0.185714, 0.214286, 0.242857, 0.271429, 0.0125, 0.0375, 0.0625, 0.0875, 0.1125, 0.1375, 0.1625, 0.1875, 0.2125, 0.2375, 0.0111111, 0.0333333, 0.0555556, 0.0777778, 0.1, 0.122222, 0.144444, 0.166667, 0.188889, 0.211111}

Promedio de peticiones pendientes para cada coeficiente:

{0.111111, 0.428571, 1., 2.33333, 9., 0.0526316, 0.176471, 0.333333, 0.538462, 0.818182, 1.22222, 1.85714, 3., 5.66667, 19., 0.0344828, 0.111111, 0.2, 0.304348, 0.428571, 0.578947, 0.764706, 1., 1.30769, 1.72727, 0.025641, 0.0810811, 0.142857, 0.212121, 0.290323, 0.37931, 0.481481, 0.6, 0.73913, 0.904762, 0.0204082, 0.0638298, 0.111111, 0.162791, 0.219512, 0.282051, 0.351351, 0.428571, 0.515152, 0.612903, 0.0169492, 0.0526316, 0.0909091, 0.132075, 0.176471, 0.22449, 0.276596, 0.333333, 0.395349, 0.463415, 0.0144928, 0.0447761, 0.0769231, 0.111111, 0.147541, 0.186441, 0.22807, 0.272727, 0.320755, 0.372549, 0.0126582, 0.038961, 0.0666667, 0.0958904, 0.126761, 0.15942, 0.19403, 0.230769, 0.269841, 0.311475, 0.011236, 0.0344828, 0.0588235, 0.0843373, 0.111111, 0.139241, 0.168831, 0.2, 0.232877, 0.267606}

Pares (coeficientes,promedio):

{{0.1, 0.111111}, {0.3, 0.428571}, {0.5, 1.}, {0.7, 2.33333}, {0.9, 9.}, {0.05, 0.0526316},

{0.15, 0.176471}, {0.25, 0.333333}, {0.35, 0.538462}, {0.45, 0.818182},

{0.55, 1.22222}, {0.65, 1.85714}, {0.75, 3.}, {0.85, 5.66667}, {0.95, 19.},

{0.0333333, 0.0344828}, {0.1, 0.111111}, {0.166667, 0.2}, {0.233333, 0.304348},

{0.3, 0.428571}, {0.366667, 0.578947}, {0.433333, 0.764706}, {0.5, 1.},

{0.566667, 1.30769}, {0.633333, 1.72727}, {0.025, 0.025641}, {0.075, 0.0810811},

{0.125, 0.142857}, {0.175, 0.212121}, {0.225, 0.290323}, {0.275, 0.37931},

{0.325, 0.481481}, {0.375, 0.6}, {0.425, 0.73913}, {0.475, 0.904762},

{0.02, 0.0204082}, {0.06, 0.0638298}, {0.1, 0.111111}, {0.14, 0.162791},

{0.18, 0.219512}, {0.22, 0.282051}, {0.26, 0.351351}, {0.3, 0.428571},

{0.34, 0.515152}, {0.38, 0.612903}, {0.0166667, 0.0169492}, {0.05, 0.0526316},

{0.0833333, 0.0909091}, {0.116667, 0.132075}, {0.15, 0.176471}, {0.183333, 0.22449},

{0.216667, 0.276596}, {0.25, 0.333333}, {0.283333, 0.395349}, {0.316667, 0.463415},

{0.0142857, 0.0144928}, {0.0428571, 0.0447761}, {0.0714286, 0.0769231}, {0.1, 0.111111},

{0.128571, 0.147541}, {0.157143, 0.186441}, {0.185714, 0.22807}, {0.214286, 0.272727},

{0.242857, 0.320755}, {0.271429, 0.372549}, {0.0125, 0.0126582}, {0.0375, 0.038961},

{0.0625, 0.0666667}, {0.0875, 0.0958904}, {0.1125, 0.126761}, {0.1375, 0.15942},

{0.1625, 0.19403}, {0.1875, 0.230769}, {0.2125, 0.269841}, {0.2375, 0.311475},

{0.0111111, 0.011236}, {0.0333333, 0.0344828}, {0.0555556, 0.0588235},

{0.0777778, 0.0843373}, {0.1, 0.111111}, {0.122222, 0.139241},

{0.144444, 0.168831}, {0.166667, 0.2}, {0.188889, 0.232877},

{0.211111, 0.267606}}

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

{Statistics‘LinearRegression‘BestFit ->

-0.639719 + 14.0473 x - 53.7404 x^2 + 58.8354 x^3

Statistics‘LinearRegression‘BestFitCoefficien ts ->

{-0.639719, 14.0473, -53.7404, 58.8354}}

Colas1pn[10,20,3]

Throw[Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.]

Colas1pn[10,90,9]

Throw[Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.]

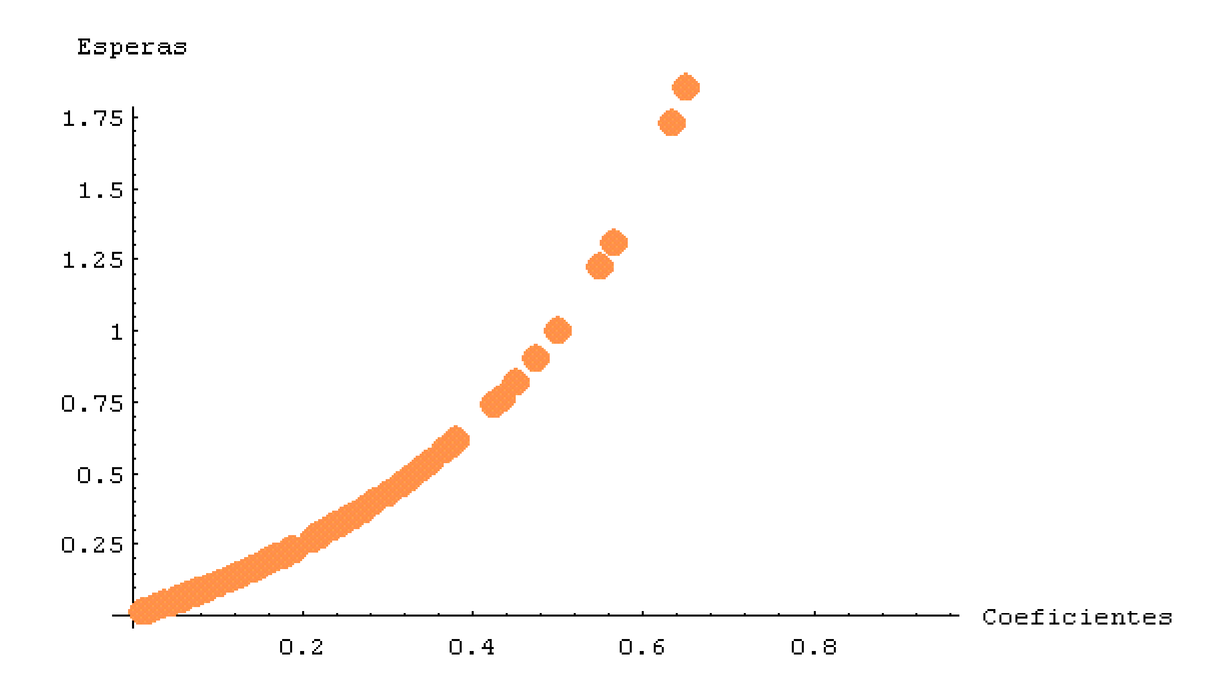


Figura 24.3: Promedio de peticiones pendientes.

Capítulo 25

Análisis del Rendimiento de un Subsistema de Disco de Varias Peticiones a la Vez con Mathematica

25.1 Datos y Ejecuciones

Contenido del archivo **Colas2en.ma:**

<<Examples‘Colas2pn‘

? Colas2pn

Colas2pn[mu,lambda,i] Análisis del rendimiento de un subsistema de disco que puede dar servicio a más de una petición a la vez.

Colocar los valores máximos para mu, lambda y el número de peticiones pendientes.

Colas2pn[20,10,3]

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806}}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.740818, 0.222245, 0.0333368, 0.00333368}}

El promedio de peticiones pendientes es el siguiente:

0.3

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.606531, 0.303265, 0.0758163, 0.0126361}}

El promedio de peticiones pendientes es el siguiente:

0.5

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.496585, 0.34761, 0.121663, 0.0283881}}

El promedio de peticiones pendientes es el siguiente:

0.7

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.40657, 0.365913, 0.164661, 0.0493982}}

El promedio de peticiones pendientes es el siguiente:

0.9

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.951229, 0.0475615, 0.00118904, 0.0000198173}}

El promedio de peticiones pendientes es el siguiente:

0.05

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.860708, 0.129106, 0.00968296, 0.000484148}}

El promedio de peticiones pendientes es el siguiente:

0.15

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.778801, 0.1947, 0.0243375, 0.00202813}}

El promedio de peticiones pendientes es el siguiente:

0.25

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.704688, 0.246641, 0.0431621, 0.00503558}}

El promedio de peticiones pendientes es el siguiente:

0.35

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.637628, 0.286933, 0.0645599, 0.00968398}}

El promedio de peticiones pendientes es el siguiente:

0.45

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45}

Promedio de peticiones pendientes para cada coeficiente:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45}

Pares (coeficientes,promedio):

{{0.1, 0.1}, {0.3, 0.3}, {0.5, 0.5}, {0.7, 0.7}, {0.9, 0.9}, {0.05, 0.05},

{0.15, 0.15}, {0.25, 0.25}, {0.35, 0.35}, {0.45, 0.45}}

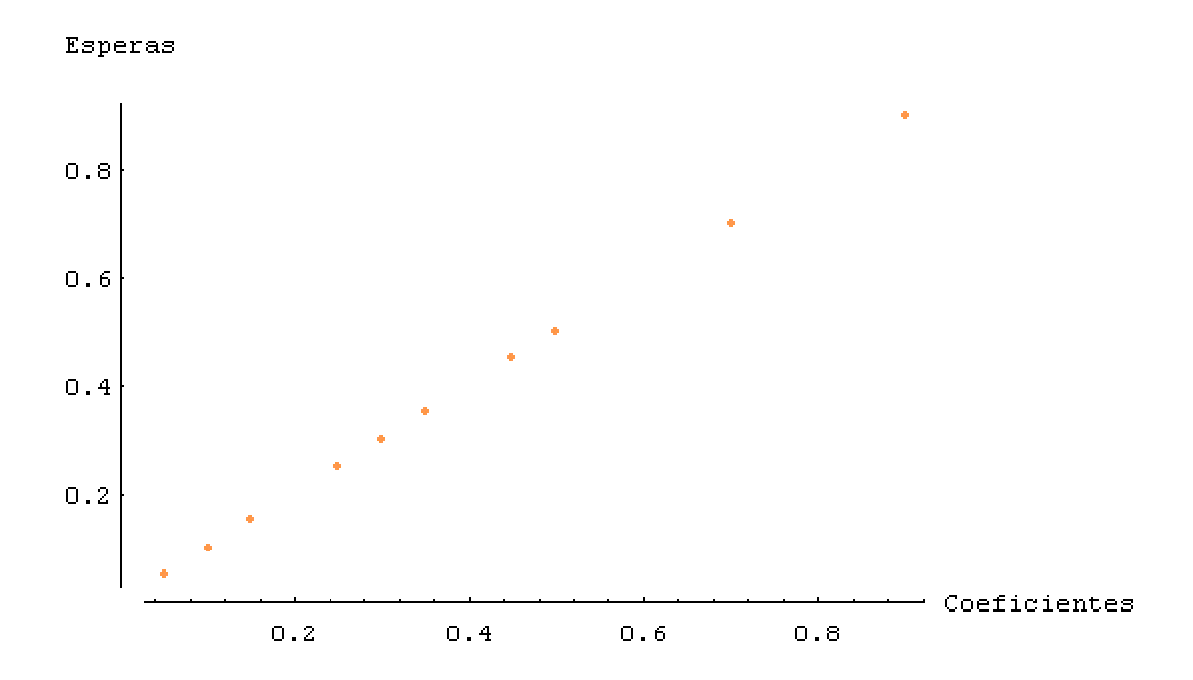


Figura 25.1: Promedio de peticiones pendientes.

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

{Statistics‘LinearRegression‘BestFit ->

-2.37982 10^{-17} + 1. x,

Statistics‘LinearRegression‘BestFitCoefficients ->

{-2.37982 10^{-17} , 1.}}

Colas2pn[40,20,7]

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806, 3.77016 10^{-6} , 7.54031 10^{-8} , 1.25672 10^{-9} , 1.79531 10^{-11} }}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.740818, 0.222245, 0.0333368, 0.00333368, 0.000250026, 0.0000150016, 7.50078 10^{-7} , 3.21462 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.3

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.606531, 0.303265, 0.0758163, 0.0126361, 0.00157951, 0.000157951, 0.0000131626, 9.40183 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.5

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.496585, 0.34761, 0.121663, 0.0283881, 0.00496792, 0.000695509, 0.0000811427, 8.11427 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.7

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.40657, 0.365913, 0.164661, 0.0493982, 0.0111146, 0.00200063,

0.000300094, 0.0000385835}}

El promedio de peticiones pendientes es el siguiente:

0.9

****

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.951229, 0.0475615, 0.00118904, 0.0000198173, 2.47716 10^{-7} , 2.47716 10^{-9} , 2.0643 10^{-11} , 1.4745 10^{-13} }}

El promedio de peticiones pendientes es el siguiente:

0.05

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.860708, 0.129106, 0.00968296, 0.000484148, 0.0000181556, 5.44667 10^{-7} , 1.36167 10^{-8} , 2.91786 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.15

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.778801, 0.1947, 0.0243375, 0.00202813, 0.000126758, $6.3379\ 10^{-6}$, $2.64079\ 10^{-7}$, $9.43139\ 10^{-9}$ }}

El promedio de peticiones pendientes es el siguiente:

0.25

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.704688, 0.246641, 0.0431621, 0.00503558, 0.000440614, 0.0000308429, $1.79917\ 10^{-6}$, $8.99586\ 10^{-8}$ }}

El promedio de peticiones pendientes es el siguiente:

0.35

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.637628, 0.286933, 0.0645599, 0.00968398, 0.00108945, 0.0000980503, $7.35377\ 10^{-6}$, $4.72742\ 10^{-7}$ }}

El promedio de peticiones pendientes es el siguiente:

0.45

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 11., 0.55}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.57695, 0.317322, 0.0872637, 0.0159983, 0.00219977, 0.000241975, 0.000022181, 1.7428 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.55

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 13., 0.65}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.522046, 0.33933, 0.110282, 0.0238945, 0.00388285, 0.000504771, 0.0000546835, 5.07775 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.65

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 15., 0.75}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.472367, 0.354275, 0.132853, 0.0332133, 0.00622749, 0.000934123,

0.000116765, 0.0000125106}}

El promedio de peticiones pendientes es el siguiente:

0.75

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 17., 0.85}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.427415, 0.363303, 0.154404, 0.0437477, 0.00929639, 0.00158039,

0.000223888, 0.0000271864}}

El promedio de peticiones pendientes es el siguiente:

0.85

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 19., 0.95}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.386741, 0.367404, 0.174517, 0.0552637, 0.0131251, 0.00249377,

0.000394847, 0.0000535864}}

El promedio de peticiones pendientes es el siguiente:

0.95

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 1., 0.0333333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.967216, 0.0322405, 0.000537342, 5.97047 10^{-6} , 4.97539 10^{-8} , 3.31693 10^{-10} , 1.84274 10^{-12} , 8.77494 10^{-15} }}

El promedio de peticiones pendientes es el siguiente:

0.0333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 3., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806, 3.77016 10^{-6} , 7.54031 10^{-8} , 1.25672 10^{-9} , 1.79531 10^{-11} }}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 5., 0.166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.846482, 0.14108, 0.0117567, 0.000653149, 0.0000272146, 9.07152 10^{-7} , 2.51987 10^{-8} , 5.99968 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.166667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 7., 0.233333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.79189, 0.184774, 0.021557, 0.00167666, 0.0000978049, 4.56423 10^{-6} , 1.77498 10^{-7} , 5.91659 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.233333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 9., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.740818, 0.222245, 0.0333368, 0.00333368, 0.000250026, 0.0000150016, 7.50078 10^{-7} , 3.21462 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.3

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 11., 0.366667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.693041, 0.254115, 0.0465877, 0.00569406, 0.000521955, 0.0000382767, 2.33913 10^{-6} , 1.22526 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.366667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 13., 0.433333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.648344, 0.280949, 0.0608723, 0.00879267, 0.000952539, 0.0000825534, 5.96219 10^{-6} , 3.69088 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.433333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 15., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.606531, 0.303265, 0.0758163, 0.0126361, 0.00157951, 0.000157951, 0.0000131626, 9.40183 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.5

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 17., 0.566667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.567414, 0.321534, 0.0911014, 0.017208, 0.00243781, 0.000276285, 0.0000260936, 2.11234 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.566667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 19., 0.633333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.530819, 0.336186, 0.106459, 0.0224746, 0.00355848, 0.000450741, 0.0000475782, 4.3047 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.633333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 1., 0.025}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.97531, 0.0243827, 0.000304784, 2.53987 10^{-6} , 1.58742 10^{-8} , 7.93709 10^{-11} , 3.30712 10^{-13} , 1.18111 10^{-15} }}

El promedio de peticiones pendientes es el siguiente:

0.025

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 3., 0.075}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.927743, 0.0695808, 0.00260928, 0.000065232, 1.2231 10^{-6} , 1.83465 10^{-8} , 2.29331 10^{-10} , 2.45712 10^{-12} }}

El promedio de peticiones pendientes es el siguiente:

0.075

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 5., 0.125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.882497, 0.110312, 0.00689451, 0.000287271, 8.97722 10^{-6} , 2.24431 10^{-7} , 4.67564 10^{-9} , 8.34935 10^{-11} }}

El promedio de peticiones pendientes es el siguiente:

0.125

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 7., 0.175}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.839457, 0.146905, 0.0128542, 0.000749827, 0.000032805, 1.14817 10^{-6} , 3.34884 10^{-8} , 8.3721 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.175

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 9., 0.225}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.798516, 0.179666, 0.0202124, 0.00151593, 0.0000852712, 3.83721 10^{-6} , 1.43895 10^{-7} , 4.6252 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.225

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 11., 0.275}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.759572, 0.208882, 0.0287213, 0.00263279, 0.000181004, 9.95523 10^{-6} , 4.56281 10^{-7} , 1.79253 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.275

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 13., 0.325}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.722527, 0.234821, 0.0381585, 0.00413383, 0.000335874, 0.0000218318, 1.18256 10^{-6} , 5.49044 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.325

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 15., 0.375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.687289, 0.257733, 0.048325, 0.00604063, 0.000566309, 0.0000424732, 2.65457 10^{-6} , 1.42209 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.375

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 17., 0.425}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.65377, 0.277852, 0.0590436, 0.00836451, 0.000888729, 0.000075542, 5.35089 10^{-6} , 3.24875 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.425

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 19., 0.475}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.621885, 0.295395, 0.0701564, 0.0111081, 0.00131909, 0.000125313, 9.92063 10^{-6} , 6.73186 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.475

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 0.0333333, 0.1, 0.166667, 0.233333, 0.3, 0.366667, 0.433333, 0.5, 0.566667, 0.633333, 0.025, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, 0.425, 0.475}

Promedio de peticiones pendientes para cada coeficiente:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 0.0333333, 0.1, 0.166667, 0.233333, 0.3, 0.366667, 0.433333, 0.5, 0.566667, 0.633333, 0.025, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, 0.425, 0.475}

Pares (coeficientes,promedio):

{{0.1, 0.1}, {0.3, 0.3}, {0.5, 0.5}, {0.7, 0.7}, {0.9, 0.9}, {0.05, 0.05}, {0.15, 0.15},

{0.25, 0.25}, {0.35, 0.35}, {0.45, 0.45}, {0.55, 0.55}, {0.65, 0.65}, {0.75, 0.75},

{0.85, 0.85}, {0.95, 0.95}, {0.0333333, 0.0333333}, {0.1, 0.1},

{0.166667, 0.166667}, {0.233333, 0.233333}, {0.3, 0.3}, {0.366667, 0.366667},

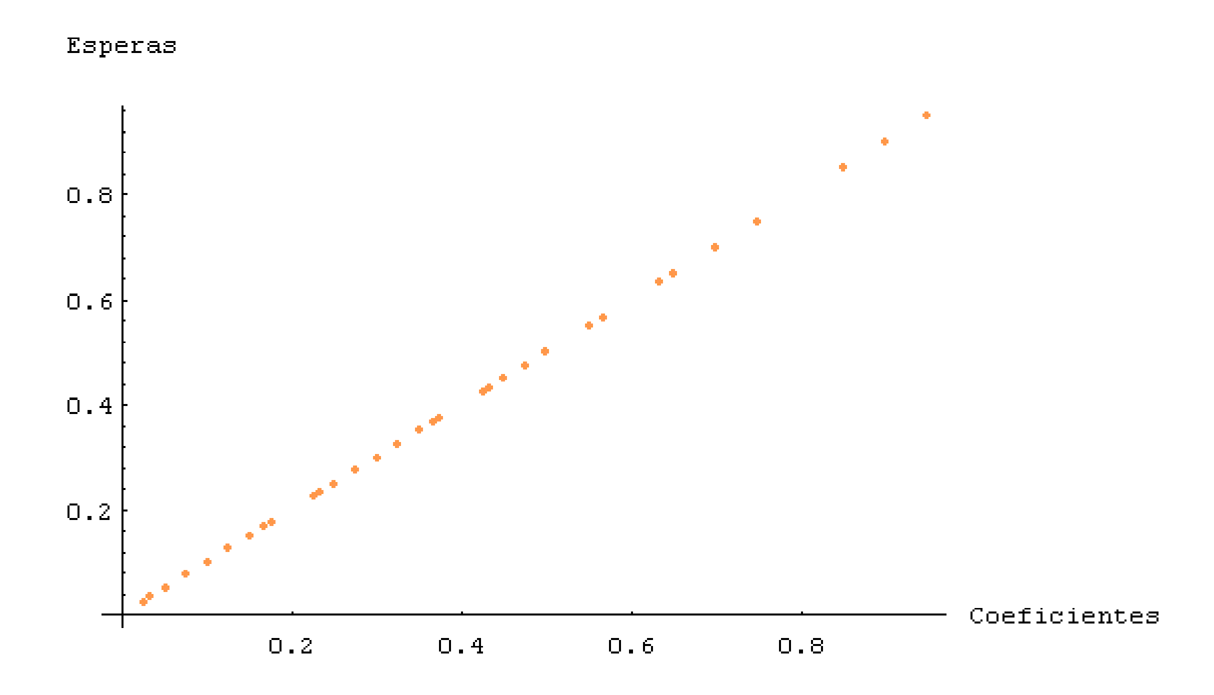


Figura 25.2: Promedio de peticiones pendientes.

{0.433333, 0.433333}, {0.5, 0.5}, {0.566667, 0.566667}, {0.633333, 0.633333},

{0.025, 0.025}, {0.075, 0.075}, {0.125, 0.125}, {0.175, 0.175}, {0.225, 0.225},

{0.275, 0.275}, {0.325, 0.325}, {0.375, 0.375}, {0.425, 0.425}, {0.475, 0.475}}

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

{Statistics‘LinearRegression‘BestFit ->

-1.33926 10^{-16} + 1. x,

Statistics‘LinearRegression‘BestFitCoefficients ->

{-1.33926 10^{-16} , 1.}}

Colas2pn[90,20,5]

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 1., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806, 3.77016 10^{-6} , 7.54031 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 3., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.740818, 0.222245, 0.0333368, 0.00333368, 0.000250026, 0.0000150016}}

El promedio de peticiones pendientes es el siguiente:

0.3

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 5., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.606531, 0.303265, 0.0758163, 0.0126361, 0.00157951, 0.000157951}}

El promedio de peticiones pendientes es el siguiente:

0.5

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 7., 0.7}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.496585, 0.34761, 0.121663, 0.0283881, 0.00496792, 0.000695509}}

El promedio de peticiones pendientes es el siguiente:

0.7

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{10., 9., 0.9}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.40657, 0.365913, 0.164661, 0.0493982, 0.0111146, 0.00200063}}

El promedio de peticiones pendientes es el siguiente:

0.9

****

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 1., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.951229, 0.0475615, 0.00118904, 0.0000198173, 2.47716 10^{-7} , 2.47716 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.05

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 3., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.860708, 0.129106, 0.00968296, 0.000484148, 0.0000181556, 5.44667 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.15

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 5., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.778801, 0.1947, 0.0243375, 0.00202813, 0.000126758, 6.3379 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.25

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 7., 0.35}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.704688, 0.246641, 0.0431621, 0.00503558, 0.000440614, 0.0000308429}}

El promedio de peticiones pendientes es el siguiente:

0.35

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 9., 0.45}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.637628, 0.286933, 0.0645599, 0.00968398, 0.00108945, 0.0000980503}}

El promedio de peticiones pendientes es el siguiente:

0.45

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 11., 0.55}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.57695, 0.317322, 0.0872637, 0.0159983, 0.00219977, 0.000241975}}

El promedio de peticiones pendientes es el siguiente:

0.55

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 13., 0.65}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.522046, 0.33933, 0.110282, 0.0238945, 0.00388285, 0.000504771}}

El promedio de peticiones pendientes es el siguiente:

0.65

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 15., 0.75}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.472367, 0.354275, 0.132853, 0.0332133, 0.00622749, 0.000934123}}

El promedio de peticiones pendientes es el siguiente:

0.75

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 17., 0.85}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.427415, 0.363303, 0.154404, 0.0437477, 0.00929639, 0.00158039}}

El promedio de peticiones pendientes es el siguiente:

0.85

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{20., 19., 0.95}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.386741, 0.367404, 0.174517, 0.0552637, 0.0131251, 0.00249377}}

El promedio de peticiones pendientes es el siguiente:

0.95

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 1., 0.0333333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.967216, 0.0322405, 0.000537342, 5.97047 10^{-6} , 4.97539 10^{-8} , 3.31693 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.0333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 3., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806, 3.77016 10^{-6} , 7.54031 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 5., 0.166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.846482, 0.14108, 0.0117567, 0.000653149, 0.0000272146, 9.07152 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.166667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 7., 0.233333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.79189, 0.184774, 0.021557, 0.00167666, 0.0000978049, 4.56423 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.233333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 9., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.740818, 0.222245, 0.0333368, 0.00333368, 0.000250026, 0.0000150016}}

El promedio de peticiones pendientes es el siguiente:

0.3

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 11., 0.366667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.693041, 0.254115, 0.0465877, 0.00569406, 0.000521955, 0.0000382767}}

El promedio de peticiones pendientes es el siguiente:

0.366667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 13., 0.433333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.648344, 0.280949, 0.0608723, 0.00879267, 0.000952539, 0.0000825534}}

El promedio de peticiones pendientes es el siguiente:

0.433333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 15., 0.5}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.606531, 0.303265, 0.0758163, 0.0126361, 0.00157951, 0.000157951}}

El promedio de peticiones pendientes es el siguiente:

0.5

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 17., 0.566667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.567414, 0.321534, 0.0911014, 0.017208, 0.00243781, 0.000276285}}

El promedio de peticiones pendientes es el siguiente:

0.566667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{30., 19., 0.633333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.530819, 0.336186, 0.106459, 0.0224746, 0.00355848, 0.000450741}}

El promedio de peticiones pendientes es el siguiente:

0.633333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 1., 0.025}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.97531, 0.0243827, 0.000304784, 2.53987 10^{-6} , 1.58742 10^{-8} , 7.93709 10^{-11} }}

El promedio de peticiones pendientes es el siguiente:

0.025

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 3., 0.075}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.927743, 0.0695808, 0.00260928, 0.000065232, 1.2231 10^{-6} , 1.83465 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.075

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 5., 0.125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.882497, 0.110312, 0.00689451, 0.000287271, 8.97722 10^{-6} , 2.24431 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.125

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 7., 0.175}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.839457, 0.146905, 0.0128542, 0.000749827, 0.000032805, 1.14817 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.175

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 9., 0.225}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.798516, 0.179666, 0.0202124, 0.00151593, 0.0000852712, 3.83721 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.225

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 11., 0.275}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.759572, 0.208882, 0.0287213, 0.00263279, 0.000181004, 9.95523 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.275

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 13., 0.325}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.722527, 0.234821, 0.0381585, 0.00413383, 0.000335874, 0.0000218318}}

El promedio de peticiones pendientes es el siguiente:

0.325

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 15., 0.375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.687289, 0.257733, 0.048325, 0.00604063, 0.000566309, 0.0000424732}}

El promedio de peticiones pendientes es el siguiente:

0.375

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 17., 0.425}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.65377, 0.277852, 0.0590436, 0.00836451, 0.000888729, 0.000075542}}

El promedio de peticiones pendientes es el siguiente:

0.425

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{40., 19., 0.475}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.621885, 0.295395, 0.0701564, 0.0111081, 0.00131909, 0.000125313}}

El promedio de peticiones pendientes es el siguiente:

0.475

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 1., 0.02}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.980199, 0.019604, 0.00019604, 1.30693 10^{-6} , 6.53466 10^{-9} , 2.61386 10^{-11} }}

El promedio de peticiones pendientes es el siguiente:

0.02

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 3., 0.06}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.941765, 0.0565059, 0.00169518, 0.0000339035, 5.08553 10^{-7} , 6.10263 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.06

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 5., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806, 3.77016 10^{-6} , 7.54031 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 7., 0.14}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.869358, 0.12171, 0.00851971, 0.000397586, 0.0000139155, 3.89635 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.14

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 9., 0.18}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.83527, 0.150349, 0.0135314, 0.000811883, 0.0000365347, 1.31525 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.18

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 11., 0.22}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.802519, 0.176554, 0.019421, 0.0014242, 0.0000783312, 3.44657 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.22

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 13., 0.26}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.771052, 0.200473, 0.0260615, 0.00225867, 0.000146813, 7.63429 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.26

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 15., 0.3}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.740818, 0.222245, 0.0333368, 0.00333368, 0.000250026, 0.0000150016}}

El promedio de peticiones pendientes es el siguiente:

0.3

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 17., 0.34}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.71177, 0.242002, 0.0411403, 0.00466257, 0.000396318, 0.0000269497}}

El promedio de peticiones pendientes es el siguiente:

0.34

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{50., 19., 0.38}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.683861, 0.259867, 0.0493748, 0.00625414, 0.000594143, 0.0000451549}}

El promedio de peticiones pendientes es el siguiente:

0.38

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 1., 0.0166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.983471, 0.0163912, 0.000136593, $7.58851\ 10^{-7}$, $3.16188\ 10^{-9}$, $1.05396\ 10^{-11}$ }}

El promedio de peticiones pendientes es el siguiente:

0.0166667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 3., 0.05}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.951229, 0.0475615, 0.00118904, 0.0000198173, 2.47716 10^{-7} , 2.47716 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.05

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 5., 0.0833333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.920044, 0.0766704, 0.0031946, 0.0000887389, 1.84873 10^{-6} , 3.08121 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.0833333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 7., 0.116667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.889882, 0.10382, 0.00605614, 0.000235517, 6.86923 10^{-6} , 1.60282 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.116667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 9., 0.15}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.860708, 0.129106, 0.00968296, 0.000484148, 0.0000181556, 5.44667 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.15

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 11., 0.183333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.832491, 0.152623, 0.0139905, 0.000854973, 0.0000391863, 1.43683 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.183333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 13., 0.216667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.805198, 0.17446, 0.0188998, 0.00136499, 0.0000739367, 3.20392 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.216667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 15., 0.25}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.778801, 0.1947, 0.0243375, 0.00202813, 0.000126758, 6.3379 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.25

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 17., 0.283333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.753269, 0.213426, 0.0302354, 0.00285556, 0.000202269, 0.0000114619}}

El promedio de peticiones pendientes es el siguiente:

0.283333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{60., 19., 0.316667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.728574, 0.230715, 0.0365299, 0.00385593, 0.000305261, 0.0000193332}}

El promedio de peticiones pendientes es el siguiente:

0.316667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 1., 0.0142857}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.985816, 0.0140831, 0.000100593, $4.79016\ 10^{-7}$, $1.71077\ 10^{-9}$, $4.88792\ 10^{-12}$ }}

El promedio de peticiones pendientes es el siguiente:

0.0142857

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 3., 0.0428571}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.958048, 0.0410592, 0.00087984, 0.0000125691, 1.34669 10^{-7} , 1.15431 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.0428571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 5., 0.0714286}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.931063, 0.0665045, 0.00237516, 0.0000565514, 1.00985 10^{-6} , 1.44264 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.0714286

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 7., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806, 3.77016 10^{-6} , 7.54031 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 9., 0.128571}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.879351, 0.113059, 0.0072681, 0.00031149, 0.0000100122, 2.57456 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.128571

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 11., 0.157143}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.854582, 0.134291, 0.0105515, 0.000552696, 0.0000217131, 6.82411 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.157143

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 13., 0.185714}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.830511, 0.154238, 0.0143221, 0.000886605, 0.0000411638, 1.52894 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.185714

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 15., 0.214286}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.807118, 0.172954, 0.0185308, 0.00132363, 0.0000709085, 3.03894 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.214286

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 17., 0.242857}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.784384, 0.190493, 0.0231313, 0.00187253, 0.00011369, 5.52207 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.242857

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{70., 19., 0.271429}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.76229, 0.206907, 0.0280803, 0.0025406, 0.000172398, 9.35872 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.271429

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 1., 0.0125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.987578, 0.0123447, 0.0000771545, 3.21477 10^{-7} , 1.00462 10^{-9} , 2.51154 10^{-12} }}

El promedio de peticiones pendientes es el siguiente:

0.0125

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 3., 0.0375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.963194, 0.0361198, 0.000677246, 8.46558 10^{-6} , 7.93648 10^{-8} , 5.95236 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.0375

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 5., 0.0625}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.939413, 0.0587133, 0.00183479, 0.0000382248, 5.97263 10^{-7} , 7.46578 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.0625

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 7., 0.0875}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.916219, 0.0801692, 0.0035074, 0.000102299, 2.23779 10^{-6} , 3.91614 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.0875

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 9., 0.1125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.893597, 0.10053, 0.0056548, 0.000212055, 5.96404 10^{-6} , 1.34191 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.1125

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 11., 0.1375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.871534, 0.119836, 0.00823872, 0.000377608, 0.0000129803, 3.56958 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.1375

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 13., 0.1625}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.850016, 0.138128, 0.0112229, 0.000607905, 0.0000246962, 8.02625 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.1625

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 15., 0.1875}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.829029, 0.155443, 0.0145728, 0.000910799, 0.0000426937, 1.60101 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.1875

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 17., 0.2125}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.80856, 0.171819, 0.0182558, 0.00129312, 0.0000686969, 2.91962 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.2125

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{80., 19., 0.2375}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.788597, 0.187292, 0.0222409, 0.00176074, 0.000104544, 4.96583 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.2375

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 1., 0.0111111}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.98895, 0.0109883, 0.0000610463, 2.26097 10^{-7} , 6.28049 10^{-10} , 1.39566 10^{-12} }}

El promedio de peticiones pendientes es el siguiente:

0.0111111

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 3., 0.0333333}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.967216, 0.0322405, 0.000537342, 5.97047 10^{-6} , 4.97539 10^{-8} , 3.31693 10^{-10} }}

El promedio de peticiones pendientes es el siguiente:

0.0333333

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 5., 0.0555556}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.945959, 0.0525533, 0.00145981, 0.0000270336, 3.75467 10^{-7} , 4.17185 10^{-9} }}

El promedio de peticiones pendientes es el siguiente:

0.0555556

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 7., 0.0777778}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.92517, 0.0719577, 0.00279835, 0.0000725499, 1.41069 10^{-6} , 2.19441 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.0777778

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 9., 0.1}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.904837, 0.0904837, 0.00452419, 0.000150806, 3.77016 10^{-6} , 7.54031 10^{-8} }}

El promedio de peticiones pendientes es el siguiente:

0.1

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 11., 0.122222}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.884952, 0.108161, 0.00660982, 0.000269289, 8.22828 10^{-6} , 2.01136 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.122222

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 13., 0.144444}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.865503, 0.125017, 0.00902901, 0.00043473, 0.0000156986, 4.53515 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.144444

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 15., 0.166667}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.846482, 0.14108, 0.0117567, 0.000653149, 0.0000272146, 9.07152 10^{-7} }}

El promedio de peticiones pendientes es el siguiente:

0.166667

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 17., 0.188889}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.827878, 0.156377, 0.0147689, 0.000929896, 0.0000439118, 1.65889 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.188889

****

Análisis del rendimiento de un subsistema de disco.

Resultados del Caso 2:

El subsistema de disco puede dar servicio a varias peticiones a la vez.

Los valores de mu, lambda y el coeficiente son los siguientes:

{90., 19., 0.211111}

La cola no crecerá indefinidamente debido a que Mu es mayor que Lambda.

Las probabilidades de tener 0,1,2,3,4,...peticiones pendientes son las siguientes:

{{0.809684, 0.170933, 0.018043, 0.00126969, 0.0000670114, 2.82937 10^{-6} }}

El promedio de peticiones pendientes es el siguiente:

0.211111

****

****

******* Resúmen final *******

****

Lista de coeficientes lambda/mu:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 0.0333333, 0.1, 0.166667, 0.233333, 0.3, 0.366667, 0.433333, 0.5, 0.566667, 0.633333, 0.025, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, 0.425, 0.475, 0.02, 0.06, 0.1, 0.14, 0.18, 0.22, 0.26, 0.3, 0.34, 0.38, 0.0166667, 0.05, 0.0833333, 0.116667, 0.15, 0.183333, 0.216667, 0.25, 0.283333, 0.316667, 0.0142857, 0.0428571, 0.0714286, 0.1, 0.128571, 0.157143, 0.185714, 0.214286, 0.242857, 0.271429, 0.0125, 0.0375, 0.0625, 0.0875, 0.1125, 0.1375, 0.1625, 0.1875, 0.2125, 0.2375, 0.0111111, 0.0333333, 0.0555556, 0.0777778, 0.1, 0.122222, 0.144444, 0.166667, 0.188889, 0.211111}

Promedio de peticiones pendientes para cada coeficiente:

{0.1, 0.3, 0.5, 0.7, 0.9, 0.05, 0.15, 0.25, 0.35, 0.45, 0.55, 0.65, 0.75, 0.85, 0.95, 0.0333333, 0.1, 0.166667, 0.233333, 0.3, 0.366667, 0.433333, 0.5, 0.566667, 0.633333, 0.025, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, 0.425, 0.475, 0.02, 0.06, 0.1, 0.14, 0.18, 0.22, 0.26, 0.3, 0.34, 0.38, 0.0166667, 0.05, 0.0833333, 0.116667, 0.15, 0.183333, 0.216667, 0.25, 0.283333, 0.316667, 0.0142857, 0.0428571, 0.0714286, 0.1, 0.128571, 0.157143, 0.185714, 0.214286, 0.242857, 0.271429, 0.0125, 0.0375, 0.0625, 0.0875, 0.1125, 0.1375, 0.1625, 0.1875, 0.2125, 0.2375, 0.0111111, 0.0333333, 0.0555556, 0.0777778, 0.1, 0.122222, 0.144444, 0.166667, 0.188889, 0.211111}

Pares (coeficientes,promedio):

{{0.1, 0.1}, {0.3, 0.3}, {0.5, 0.5}, {0.7, 0.7}, {0.9, 0.9}, {0.05, 0.05}, {0.15, 0.15},

{0.25, 0.25}, {0.35, 0.35}, {0.45, 0.45}, {0.55, 0.55}, {0.65, 0.65}, {0.75, 0.75},

{0.85, 0.85}, {0.95, 0.95}, {0.0333333, 0.0333333}, {0.1, 0.1},

{0.166667, 0.166667}, {0.233333, 0.233333}, {0.3, 0.3}, {0.366667, 0.366667},

{0.433333, 0.433333}, {0.5, 0.5}, {0.566667, 0.566667}, {0.633333, 0.633333},

{0.025, 0.025}, {0.075, 0.075}, {0.125, 0.125}, {0.175, 0.175}, {0.225, 0.225},

{0.275, 0.275}, {0.325, 0.325}, {0.375, 0.375}, {0.425, 0.425},

{0.475, 0.475}, {0.02, 0.02}, {0.06, 0.06}, {0.1, 0.1}, {0.14, 0.14}, {0.18, 0.18},

{0.22, 0.22}, {0.26, 0.26}, {0.3, 0.3}, {0.34, 0.34}, {0.38, 0.38},

{0.0166667, 0.0166667}, {0.05, 0.05}, {0.0833333, 0.0833333}, {0.116667, 0.116667},

{0.15, 0.15}, {0.183333, 0.183333}, {0.216667, 0.216667}, {0.25, 0.25},

{0.283333, 0.283333}, {0.316667, 0.316667}, {0.0142857, 0.0142857},

{0.0428571, 0.0428571}, {0.0714286, 0.0714286}, {0.1, 0.1}, {0.128571, 0.128571},

{0.157143, 0.157143}, {0.185714, 0.185714}, {0.214286, 0.214286},

{0.242857, 0.242857}, {0.271429, 0.271429}, {0.0125, 0.0125}, {0.0375, 0.0375},

{0.0625, 0.0625}, {0.0875, 0.0875}, {0.1125, 0.1125}, {0.1375, 0.1375},

{0.1625, 0.1625}, {0.1875, 0.1875}, {0.2125, 0.2125}, {0.2375, 0.2375},

{0.0111111, 0.0111111}, {0.0333333, 0.0333333}, {0.0555556, 0.0555556},

{0.0777778, 0.0777778}, {0.1, 0.1}, {0.122222, 0.122222}, {0.144444, 0.144444},

{0.166667, 0.166667}, {0.188889, 0.188889}, {0.211111, 0.211111}}

Gráfico de promedio de peticiones pendientes para cada coeficiente y Análisis de regresión lineal:

{Statistics‘LinearRegression‘BestFit ->

-2.07028 10^{-16} + 1. x,

Statistics‘LinearRegression‘BestFitCoefficients ->

{-2.07028 10^{-16}, 1.}}

Colas2pn[10,20,3]

Throw[Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.]

Colas2pn[10,90,9]

Throw[Para que la cola no crezca indefinidamente debe ser Mu mayor que Lambda.]

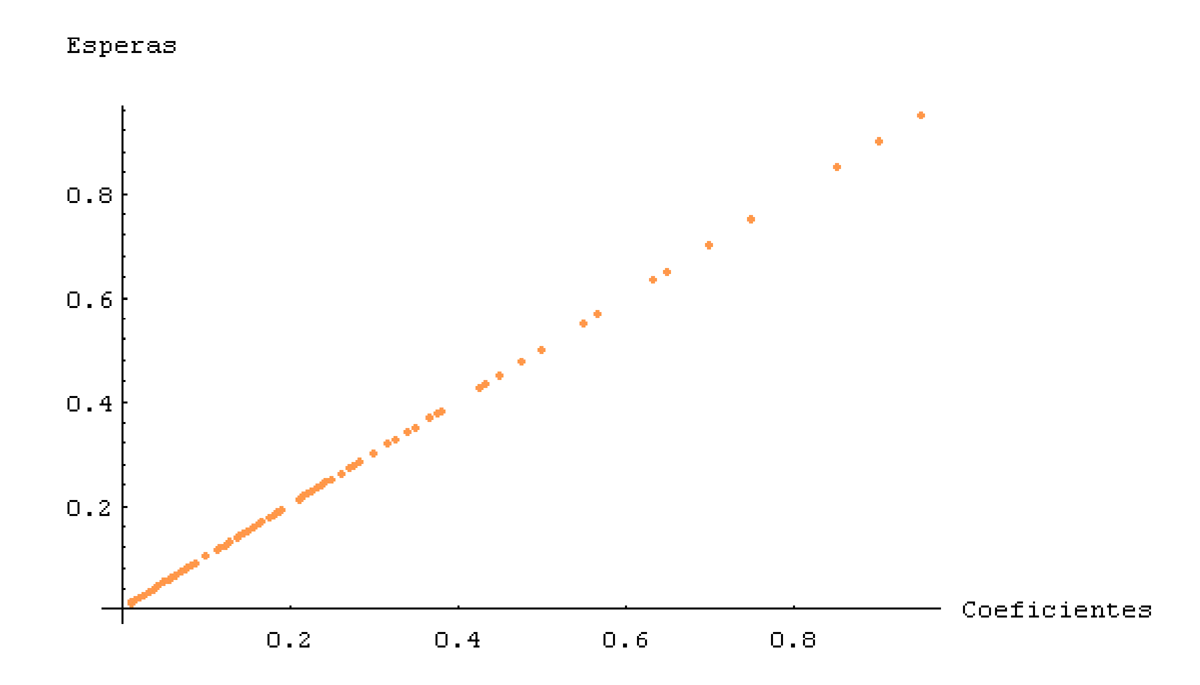


Figura 25.3: Promedio de peticiones pendientes.

Capítulo 26

Optimización de Operaciones de Búsqueda en Disco con Redes Neuronales

26.1 Datos y Ejecuciones

Contenido del archivo **Colas3en.ma:**

<<Examples‘Colas3pn‘

? Colas3pn

Colas3pn[cildisc,totpetic,numpetic]

Cálculo del número de cilindros del disco recorridos por el mecanismo de acceso en las operaciones de búsqueda de los cilindros requeridos según las peticiones de entrada / salida que precisan los procesos; generación del archivo de resultados correspondiente.

Colocar los valores para cildisc, totpetic y numpetic.

Colas3pn[40,5,100]

Se sugiere que numpetic sea al menos 30 veces mayor que totpetic.

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 1}

La posición inicial del mecanismo de acceso es la siguiente:

23

La lista aleatoria de peticiones es la siguiente:

{16, 13, 9, 31, 39}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{44, 44, 46}

El algoritmo más eficiente es:

FCFS

Imagen del registro grabado:

{23, {16, 13, 9, 31, 39}, {44, 44, 46}, 1, 0, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 2}

La posición inicial del mecanismo de acceso es la siguiente:

14

La lista aleatoria de peticiones es la siguiente:

{29, 13, 28, 26, 39}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{61, 27, 51}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{14, {29, 13, 28, 26, 39}, {61, 27, 51}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 3}

La posición inicial del mecanismo de acceso es la siguiente:

21

La lista aleatoria de peticiones es la siguiente:

{6, 38, 23, 4, 31}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{108, 51, 51}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{21, {6, 38, 23, 4, 31}, {108, 51, 51}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 4}

La posición inicial del mecanismo de acceso es la siguiente:

5

La lista aleatoria de peticiones es la siguiente:

{34, 13, 15, 20, 30}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{67, 29, 29}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{5, {34, 13, 15, 20, 30}, {67, 29, 29}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 5}

La posición inicial del mecanismo de acceso es la siguiente:

32

La lista aleatoria de peticiones es la siguiente:

{21, 28, 33, 12, 31}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{63, 22, 22}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{32, {21, 28, 33, 12, 31}, {63, 22, 22}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 6}

La posición inicial del mecanismo de acceso es la siguiente:

20

La lista aleatoria de peticiones es la siguiente:

{40, 1, 20, 2, 34}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{128, 59, 59}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{20, {40, 1, 20, 2, 34}, {128, 59, 59}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 7}

La posición inicial del mecanismo de acceso es la siguiente:

24

La lista aleatoria de peticiones es la siguiente:

{23, 3, 7, 27, 27}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{45, 29, 27}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{24, {23, 3, 7, 27, 27}, {45, 29, 27}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 8}

La posición inicial del mecanismo de acceso es la siguiente:

10

La lista aleatoria de peticiones es la siguiente:

{23, 26, 8, 18, 32}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{58, 26, 46}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{10, {23, 26, 8, 18, 32}, {58, 26, 46}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 9}

La posición inicial del mecanismo de acceso es la siguiente:

23

La lista aleatoria de peticiones es la siguiente:

{3, 17, 33, 40, 3}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{94, 57, 54}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{23, {3, 17, 33, 40, 3}, {94, 57, 54}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 10}

La posición inicial del mecanismo de acceso es la siguiente:

11

La lista aleatoria de peticiones es la siguiente:

{30, 27, 2, 30, 7}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{98, 37, 47}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{11, {30, 27, 2, 30, 7}, {98, 37, 47}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 11}

La posición inicial del mecanismo de acceso es la siguiente:

6

La lista aleatoria de peticiones es la siguiente:

{13, 27, 3, 39, 10}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{110, 39, 69}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{6, {13, 27, 3, 39, 10}, {110, 39, 69}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 12}

La posición inicial del mecanismo de acceso es la siguiente:

40

La lista aleatoria de peticiones es la siguiente:

{10, 31, 31, 11, 26}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{86, 30, 30}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{40, {10, 31, 31, 11, 26}, {86, 30, 30}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 13}

La posición inicial del mecanismo de acceso es la siguiente:

29

La lista aleatoria de peticiones es la siguiente:

{13, 15, 7, 8, 8}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{27, 22, 22}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{29, {13, 15, 7, 8, 8}, {27, 22, 22}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 14}

La posición inicial del mecanismo de acceso es la siguiente:

3

La lista aleatoria de peticiones es la siguiente:

{9, 29, 3, 35, 35}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{84, 32, 32}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{3, {9, 29, 3, 35, 35}, {84, 32, 32}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 15}

La posición inicial del mecanismo de acceso es la siguiente:

38

La lista aleatoria de peticiones es la siguiente:

{23, 31, 34, 2, 2}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{58, 36, 36}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{38, {23, 31, 34, 2, 2}, {58, 36, 36}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 16}

La posición inicial del mecanismo de acceso es la siguiente:

23

La lista aleatoria de peticiones es la siguiente:

{21, 37, 21, 18, 25}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{44, 24, 33}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{23, {21, 37, 21, 18, 25}, {44, 24, 33}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 17}

La posición inicial del mecanismo de acceso es la siguiente:

4

La lista aleatoria de peticiones es la siguiente:

{7, 33, 26, 28, 1}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{65, 41, 61}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{4, {7, 33, 26, 28, 1}, {65, 41, 61}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 18}

La posición inicial del mecanismo de acceso es la siguiente:

7

La lista aleatoria de peticiones es la siguiente:

{13, 13, 11, 2, 31}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{46, 46, 53}

El algoritmo más eficiente es:

FCFS

Imagen del registro grabado:

{7, {13, 13, 11, 2, 31}, {46, 46, 53}, 1, 0, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 19}

La posición inicial del mecanismo de acceso es la siguiente:

32

La lista aleatoria de peticiones es la siguiente:

{26, 5, 25, 1, 4}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{74, 31, 31}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{32, {26, 5, 25, 1, 4}, {74, 31, 31}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 20}

La posición inicial del mecanismo de acceso es la siguiente:

18

La lista aleatoria de peticiones es la siguiente:

{12, 3, 40, 11, 40}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{110, 52, 59}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{18, {12, 3, 40, 11, 40}, {110, 52, 59}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 21}

La posición inicial del mecanismo de acceso es la siguiente:

9

La lista aleatoria de peticiones es la siguiente:

{36, 28, 36, 3, 25}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{98, 39, 60}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{9, {36, 28, 36, 3, 25}, {98, 39, 60}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 22}

La posición inicial del mecanismo de acceso es la siguiente:

36

La lista aleatoria de peticiones es la siguiente:

{14, 22, 34, 16, 9}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{67, 27, 27}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{36, {14, 22, 34, 16, 9}, {67, 27, 27}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 23}

La posición inicial del mecanismo de acceso es la siguiente:

7

La lista aleatoria de peticiones es la siguiente:

{17, 7, 11, 23, 28}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{41, 21, 21}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{7, {17, 7, 11, 23, 28}, {41, 21, 21}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 24}

La posición inicial del mecanismo de acceso es la siguiente:

5

La lista aleatoria de peticiones es la siguiente:

{9, 26, 40, 36, 30}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{45, 35, 35}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{5, {9, 26, 40, 36, 30}, {45, 35, 35}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 25}

La posición inicial del mecanismo de acceso es la siguiente:

29

La lista aleatoria de peticiones es la siguiente:

{18, 6, 17, 27, 9}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{62, 23, 23}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{29, {18, 6, 17, 27, 9}, {62, 23, 23}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 26}

La posición inicial del mecanismo de acceso es la siguiente:

24

La lista aleatoria de peticiones es la siguiente:

{25, 4, 33, 6, 26}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{98, 38, 38}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{24, {25, 4, 33, 6, 26}, {98, 38, 38}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 27}

La posición inicial del mecanismo de acceso es la siguiente:

40

La lista aleatoria de peticiones es la siguiente:

{27, 3, 3, 30, 11}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{83, 37, 37}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{40, {27, 3, 3, 30, 11}, {83, 37, 37}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 28}

La posición inicial del mecanismo de acceso es la siguiente:

1

La lista aleatoria de peticiones es la siguiente:

{39, 36, 10, 28, 21}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{92, 38, 38}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{1, {39, 36, 10, 28, 21}, {92, 38, 38}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 29}

La posición inicial del mecanismo de acceso es la siguiente:

15

La lista aleatoria de peticiones es la siguiente:

{6, 16, 19, 3, 22}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{57, 26, 26}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{15, {6, 16, 19, 3, 22}, {57, 26, 26}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 30}

La posición inicial del mecanismo de acceso es la siguiente:

6

La lista aleatoria de peticiones es la siguiente:

{4, 20, 11, 40, 9}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{87, 38, 70}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{6, {4, 20, 11, 40, 9}, {87, 38, 70}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 31}

La posición inicial del mecanismo de acceso es la siguiente:

8

La lista aleatoria de peticiones es la siguiente:

{38, 40, 4, 40, 37}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{107, 40, 68}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{8, {38, 40, 4, 40, 37}, {107, 40, 68}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 32}

La posición inicial del mecanismo de acceso es la siguiente:

5

La lista aleatoria de peticiones es la siguiente:

{32, 36, 29, 22, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{48, 31, 31}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{5, {32, 36, 29, 22, 19}, {48, 31, 31}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 33}

La posición inicial del mecanismo de acceso es la siguiente:

22

La lista aleatoria de peticiones es la siguiente:

{13, 34, 15, 11, 29}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{71, 34, 35}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{22, {13, 34, 15, 11, 29}, {71, 34, 35}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 34}

La posición inicial del mecanismo de acceso es la siguiente:

6

La lista aleatoria de peticiones es la siguiente:

{27, 16, 3, 18, 29}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{71, 29, 49}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{6, {27, 16, 3, 18, 29}, {71, 29, 49}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 35}

La posición inicial del mecanismo de acceso es la siguiente:

3

La lista aleatoria de peticiones es la siguiente:

{3, 22, 29, 8, 37}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{76, 34, 34}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{3, {3, 22, 29, 8, 37}, {76, 34, 34}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 36}

La posición inicial del mecanismo de acceso es la siguiente:

27

La lista aleatoria de peticiones es la siguiente:

{12, 3, 11, 8, 16}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{43, 24, 24}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{27, {12, 3, 11, 8, 16}, {43, 24, 24}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 37}

La posición inicial del mecanismo de acceso es la siguiente:

36

La lista aleatoria de peticiones es la siguiente:

{6, 12, 17, 9, 16}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{56, 30, 30}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{36, {6, 12, 17, 9, 16}, {56, 30, 30}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 38}

La posición inicial del mecanismo de acceso es la siguiente:

4

La lista aleatoria de peticiones es la siguiente:

{10, 18, 2, 26, 24}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{56, 26, 46}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{4, {10, 18, 2, 26, 24}, {56, 26, 46}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 39}

La posición inicial del mecanismo de acceso es la siguiente:

2

La lista aleatoria de peticiones es la siguiente:

{16, 4, 29, 15, 39}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{89, 37, 37}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{2, {16, 4, 29, 15, 39}, {89, 37, 37}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 40}

La posición inicial del mecanismo de acceso es la siguiente:

10

La lista aleatoria de peticiones es la siguiente:

{6, 9, 23, 40, 17}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{61, 38, 64}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{10, {6, 9, 23, 40, 17}, {61, 38, 64}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 41}

La posición inicial del mecanismo de acceso es la siguiente:

4

La lista aleatoria de peticiones es la siguiente:

{35, 36, 10, 27, 40}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{88, 36, 36}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{4, {35, 36, 10, 27, 40}, {88, 36, 36}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 42}

La posición inicial del mecanismo de acceso es la siguiente:

33

La lista aleatoria de peticiones es la siguiente:

{6, 35, 26, 21, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{72, 31, 31}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{33, {6, 35, 26, 21, 19}, {72, 31, 31}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 43}

La posición inicial del mecanismo de acceso es la siguiente:

40

La lista aleatoria de peticiones es la siguiente:

{36, 10, 22, 15, 24}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{58, 30, 30}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{40, {36, 10, 22, 15, 24}, {58, 30, 30}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 44}

La posición inicial del mecanismo de acceso es la siguiente:

11

La lista aleatoria de peticiones es la siguiente:

{4, 15, 15, 19, 20}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{23, 25, 25}

El algoritmo más eficiente es:

FCFS

Imagen del registro grabado:

{11, {4, 15, 15, 19, 20}, {23, 25, 25}, 1, 0, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 45}

La posición inicial del mecanismo de acceso es la siguiente:

38

La lista aleatoria de peticiones es la siguiente:

{14, 4, 19, 16, 26}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{62, 34, 34}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{38, {14, 4, 19, 16, 26}, {62, 34, 34}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 46}

La posición inicial del mecanismo de acceso es la siguiente:

21

La lista aleatoria de peticiones es la siguiente:

{38, 14, 35, 6, 32}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{117, 47, 49}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{21, {38, 14, 35, 6, 32}, {117, 47, 49}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 47}

La posición inicial del mecanismo de acceso es la siguiente:

32

La lista aleatoria de peticiones es la siguiente:

{13, 37, 6, 1, 7}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{85, 41, 41}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{32, {13, 37, 6, 1, 7}, {85, 41, 41}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 48}

La posición inicial del mecanismo de acceso es la siguiente:

8

La lista aleatoria de peticiones es la siguiente:

{8, 26, 31, 11, 20}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{52, 23, 23}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{8, {8, 26, 31, 11, 20}, {52, 23, 23}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 49}

La posición inicial del mecanismo de acceso es la siguiente:

12

La lista aleatoria de peticiones es la siguiente:

{17, 18, 24, 7, 1}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{35, 35, 35}

El algoritmo más eficiente es:

FCFS

Imagen del registro grabado:

{12, {17, 18, 24, 7, 1}, {35, 35, 35}, 1, 0, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 50}

La posición inicial del mecanismo de acceso es la siguiente:

40

La lista aleatoria de peticiones es la siguiente:

{32, 28, 24, 3, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{53, 37, 37}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{40, {32, 28, 24, 3, 19}, {53, 37, 37}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 51}

La posición inicial del mecanismo de acceso es la siguiente:

18

La lista aleatoria de peticiones es la siguiente:

{34, 11, 12, 30, 24}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{64, 30, 39}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{18, {34, 11, 12, 30, 24}, {64, 30, 39}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 52}

La posición inicial del mecanismo de acceso es la siguiente:

1

La lista aleatoria de peticiones es la siguiente:

{39, 33, 17, 22, 9}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{78, 38, 38}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{1, {39, 33, 17, 22, 9}, {78, 38, 38}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 53}

La posición inicial del mecanismo de acceso es la siguiente:

24

La lista aleatoria de peticiones es la siguiente:

{20, 1, 16, 2, 17}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{67, 23, 23}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{24, {20, 1, 16, 2, 17}, {67, 23, 23}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 54}

La posición inicial del mecanismo de acceso es la siguiente:

36

La lista aleatoria de peticiones es la siguiente:

{11, 4, 36, 31, 12}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{88, 32, 32}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{36, {11, 4, 36, 31, 12}, {88, 32, 32}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 55}

La posición inicial del mecanismo de acceso es la siguiente:

15

La lista aleatoria de peticiones es la siguiente:

{14, 26, 8, 33, 39}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{62, 38, 55}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{15, {14, 26, 8, 33, 39}, {62, 38, 55}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 56}

La posición inicial del mecanismo de acceso es la siguiente:

3

La lista aleatoria de peticiones es la siguiente:

{19, 30, 6, 18, 4}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{77, 27, 27}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{3, {19, 30, 6, 18, 4}, {77, 27, 27}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 57}

La posición inicial del mecanismo de acceso es la siguiente:

34

La lista aleatoria de peticiones es la siguiente:

{21, 15, 4, 28, 7}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{75, 30, 30}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{34, {21, 15, 4, 28, 7}, {75, 30, 30}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 58}

La posición inicial del mecanismo de acceso es la siguiente:

11

La lista aleatoria de peticiones es la siguiente:

{6, 37, 28, 14, 20}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{65, 57, 57}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{11, {6, 37, 28, 14, 20}, {65, 57, 57}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 59}

La posición inicial del mecanismo de acceso es la siguiente:

16

La lista aleatoria de peticiones es la siguiente:

{30, 3, 9, 40, 30}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{88, 50, 61}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{16, {30, 3, 9, 40, 30}, {88, 50, 61}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 60}

La posición inicial del mecanismo de acceso es la siguiente:

19

La lista aleatoria de peticiones es la siguiente:

{12, 23, 7, 15, 14}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{43, 20, 20}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{19, {12, 23, 7, 15, 14}, {43, 20, 20}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 61}

La posición inicial del mecanismo de acceso es la siguiente:

31

La lista aleatoria de peticiones es la siguiente:

{20, 32, 36, 9, 33}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{78, 32, 32}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{31, {20, 32, 36, 9, 33}, {78, 32, 32}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 62}

La posición inicial del mecanismo de acceso es la siguiente:

33

La lista aleatoria de peticiones es la siguiente:

{9, 5, 1, 22, 38}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{69, 42, 42}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{33, {9, 5, 1, 22, 38}, {69, 42, 42}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 63}

La posición inicial del mecanismo de acceso es la siguiente:

14

La lista aleatoria de peticiones es la siguiente:

{27, 36, 9, 33, 25}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{81, 32, 49}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{14, {27, 36, 9, 33, 25}, {81, 32, 49}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 64}

La posición inicial del mecanismo de acceso es la siguiente:

31

La lista aleatoria de peticiones es la siguiente:

{13, 12, 16, 32, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{52, 21, 21}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{31, {13, 12, 16, 32, 19}, {52, 21, 21}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 65}

La posición inicial del mecanismo de acceso es la siguiente:

34

La lista aleatoria de peticiones es la siguiente:

{13, 38, 9, 17, 32}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{98, 37, 33}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{34, {13, 38, 9, 17, 32}, {98, 37, 33}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 66}

La posición inicial del mecanismo de acceso es la siguiente:

18

La lista aleatoria de peticiones es la siguiente:

{24, 8, 2, 11, 36}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{62, 52, 52}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{18, {24, 8, 2, 11, 36}, {62, 52, 52}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 67}

La posición inicial del mecanismo de acceso es la siguiente:

8

La lista aleatoria de peticiones es la siguiente:

{25, 13, 27, 8, 35}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{89, 27, 27}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{8, {25, 13, 27, 8, 35}, {89, 27, 27}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 68}

La posición inicial del mecanismo de acceso es la siguiente:

13

La lista aleatoria de peticiones es la siguiente:

{13, 7, 31, 32, 31}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{32, 31, 44}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{13, {13, 7, 31, 32, 31}, {32, 31, 44}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 69}

La posición inicial del mecanismo de acceso es la siguiente:

21

La lista aleatoria de peticiones es la siguiente:

{21, 7, 9, 12, 23}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{30, 18, 18}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{21, {21, 7, 9, 12, 23}, {30, 18, 18}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 70}

La posición inicial del mecanismo de acceso es la siguiente:

15

La lista aleatoria de peticiones es la siguiente:

{10, 22, 12, 25, 23}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{42, 20, 25}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{15, {10, 22, 12, 25, 23}, {42, 20, 25}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 71}

La posición inicial del mecanismo de acceso es la siguiente:

3

La lista aleatoria de peticiones es la siguiente:

{29, 8, 19, 37, 12}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{101, 34, 34}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{3, {29, 8, 19, 37, 12}, {101, 34, 34}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 72}

La posición inicial del mecanismo de acceso es la siguiente:

3

La lista aleatoria de peticiones es la siguiente:

{19, 3, 12, 5, 32}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{75, 29, 29}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{3, {19, 3, 12, 5, 32}, {75, 29, 29}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 73}

La posición inicial del mecanismo de acceso es la siguiente:

10

La lista aleatoria de peticiones es la siguiente:

{4, 21, 30, 3, 33}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{89, 37, 53}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{10, {4, 21, 30, 3, 33}, {89, 37, 53}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 74}

La posición inicial del mecanismo de acceso es la siguiente:

22

La lista aleatoria de peticiones es la siguiente:

{38, 10, 20, 22, 24}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{58, 48, 44}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{22, {38, 10, 20, 22, 24}, {58, 48, 44}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 75}

La posición inicial del mecanismo de acceso es la siguiente:

29

La lista aleatoria de peticiones es la siguiente:

{39, 9, 26, 29, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{70, 50, 40}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{29, {39, 9, 26, 29, 19}, {70, 50, 40}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 76}

La posición inicial del mecanismo de acceso es la siguiente:

23

La lista aleatoria de peticiones es la siguiente:

{3, 27, 8, 12, 34}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{89, 42, 42}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{23, {3, 27, 8, 12, 34}, {89, 42, 42}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 77}

La posición inicial del mecanismo de acceso es la siguiente:

14

La lista aleatoria de peticiones es la siguiente:

{34, 38, 11, 40, 33}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{87, 32, 55}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{14, {34, 38, 11, 40, 33}, {87, 32, 55}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 78}

La posición inicial del mecanismo de acceso es la siguiente:

26

La lista aleatoria de peticiones es la siguiente:

{19, 26, 35, 12, 25}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{59, 37, 32}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{26, {19, 26, 35, 12, 25}, {59, 37, 32}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 79}

La posición inicial del mecanismo de acceso es la siguiente:

31

La lista aleatoria de peticiones es la siguiente:

{22, 8, 23, 13, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{54, 23, 23}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{31, {22, 8, 23, 13, 19}, {54, 23, 23}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 80}

La posición inicial del mecanismo de acceso es la siguiente:

33

La lista aleatoria de peticiones es la siguiente:

{8, 2, 39, 5, 36}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{133, 43, 43}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{33, {8, 2, 39, 5, 36}, {133, 43, 43}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 81}

La posición inicial del mecanismo de acceso es la siguiente:

23

La lista aleatoria de peticiones es la siguiente:

{16, 34, 3, 39, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{112, 56, 52}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{23, {16, 34, 3, 39, 19}, {112, 56, 52}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 82}

La posición inicial del mecanismo de acceso es la siguiente:

30

La lista aleatoria de peticiones es la siguiente:

{25, 18, 37, 1, 21}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{87, 65, 43}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{30, {25, 18, 37, 1, 21}, {87, 65, 43}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 83}

La posición inicial del mecanismo de acceso es la siguiente:

17

La lista aleatoria de peticiones es la siguiente:

{12, 21, 18, 27, 20}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{33, 25, 25}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{17, {12, 21, 18, 27, 20}, {33, 25, 25}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 84}

La posición inicial del mecanismo de acceso es la siguiente:

9

La lista aleatoria de peticiones es la siguiente:

{31, 2, 17, 38, 30}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{95, 43, 65}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{9, {31, 2, 17, 38, 30}, {95, 43, 65}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 85}

La posición inicial del mecanismo de acceso es la siguiente:

10

La lista aleatoria de peticiones es la siguiente:

{21, 36, 40, 1, 6}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{74, 48, 69}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{10, {21, 36, 40, 1, 6}, {74, 48, 69}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 86}

La posición inicial del mecanismo de acceso es la siguiente:

39

La lista aleatoria de peticiones es la siguiente:

{17, 30, 25, 18, 38}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{67, 22, 22}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{39, {17, 30, 25, 18, 38}, {67, 22, 22}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 87}

La posición inicial del mecanismo de acceso es la siguiente:

32

La lista aleatoria de peticiones es la siguiente:

{37, 19, 18, 14, 28}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{42, 36, 28}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{32, {37, 19, 18, 14, 28}, {42, 36, 28}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 88}

La posición inicial del mecanismo de acceso es la siguiente:

11

La lista aleatoria de peticiones es la siguiente:

{28, 21, 22, 35, 36}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{39, 25, 25}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{11, {28, 21, 22, 35, 36}, {39, 25, 25}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 89}

La posición inicial del mecanismo de acceso es la siguiente:

21

La lista aleatoria de peticiones es la siguiente:

{23, 19, 28, 20, 10}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{33, 29, 25}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{21, {23, 19, 28, 20, 10}, {33, 29, 25}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 90}

La posición inicial del mecanismo de acceso es la siguiente:

6

La lista aleatoria de peticiones es la siguiente:

{38, 24, 35, 26, 19}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{73, 32, 32}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{6, {38, 24, 35, 26, 19}, {73, 32, 32}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 91}

La posición inicial del mecanismo de acceso es la siguiente:

12

La lista aleatoria de peticiones es la siguiente:

{14, 34, 35, 10, 11}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{49, 27, 48}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{12, {14, 34, 35, 10, 11}, {49, 27, 48}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 92}

La posición inicial del mecanismo de acceso es la siguiente:

20

La lista aleatoria de peticiones es la siguiente:

{10, 35, 28, 3, 36}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{100, 49, 49}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{20, {10, 35, 28, 3, 36}, {100, 49, 49}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 93}

La posición inicial del mecanismo de acceso es la siguiente:

25

La lista aleatoria de peticiones es la siguiente:

{38, 35, 1, 2, 3}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{52, 50, 50}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{25, {38, 35, 1, 2, 3}, {52, 50, 50}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 94}

La posición inicial del mecanismo de acceso es la siguiente:

22

La lista aleatoria de peticiones es la siguiente:

{1, 2, 29, 12, 5}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{73, 35, 35}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{22, {1, 2, 29, 12, 5}, {73, 35, 35}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 95}

La posición inicial del mecanismo de acceso es la siguiente:

36

La lista aleatoria de peticiones es la siguiente:

{17, 13, 34, 15, 39}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{87, 33, 29}

El algoritmo más eficiente es:

SCAN

Imagen del registro grabado:

{36, {17, 13, 34, 15, 39}, {87, 33, 29}, 0, 0, 1}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 96}

La posición inicial del mecanismo de acceso es la siguiente:

38

La lista aleatoria de peticiones es la siguiente:

{10, 19, 18, 23, 37}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{57, 28, 28}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{38, {10, 19, 18, 23, 37}, {57, 28, 28}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 97}

La posición inicial del mecanismo de acceso es la siguiente:

7

La lista aleatoria de peticiones es la siguiente:

{36, 37, 1, 17, 17}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{82, 42, 66}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{7, {36, 37, 1, 17, 17}, {82, 42, 66}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 98}

La posición inicial del mecanismo de acceso es la siguiente:

40

La lista aleatoria de peticiones es la siguiente:

{31, 28, 28, 17, 26}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{32, 23, 23}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{40, {31, 28, 28, 17, 26}, {32, 23, 23}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 99}

La posición inicial del mecanismo de acceso es la siguiente:

36

La lista aleatoria de peticiones es la siguiente:

{34, 7, 24, 29, 17}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{63, 29, 29}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{36, {34, 7, 24, 29, 17}, {63, 29, 29}, 0, 1, 0}

****

Cálculo de movimientos del mecanismo de acceso según tres algoritmos de búsqueda.

Los valores de cildisc, totpetic y numpetic son los siguientes:

{40, 5, 100}

La posición inicial del mecanismo de acceso es la siguiente:

34

La lista aleatoria de peticiones es la siguiente:

{35, 32, 14, 12, 28}

El número de movimientos del mecanismo de acceso sería el siguiente para FCFS, SSF y SCAN:

{40, 24, 24}

El algoritmo más eficiente es:

SSF

Imagen del registro grabado:

{34, {35, 32, 14, 12, 28}, {40, 24, 24}, 0, 1, 0}

****

****

** Archivo de datos grabado como Colas3fn.txt **

****

Contenido del archivo **Colas3fn.dat:**

14, 15, 2, 20, 13, 7 , 0, 0, 1

37, 9, 11, 23, 32, 36 , 0, 1, 0

6, 30, 31, 15, 26, 11 , 0, 1, 0

11, 25, 38, 34, 24, 26 , 0, 1, 0

5, 24, 25, 27, 32, 9 , 0, 1, 0

31, 18, 32, 4, 35, 20 , 0, 1, 0

38, 20, 27, 4, 24, 22 , 0, 1, 0

39, 27, 4, 30, 6, 27 , 0, 1, 0

12, 28, 33, 2, 40, 17 , 0, 1, 0

12, 24, 19, 12, 15, 19 , 0, 1, 0

40, 4, 6, 24, 20, 24 , 0, 1, 0

7, 7, 17, 32, 13, 25 , 0, 1, 0

29, 16, 2, 38, 36, 5 , 0, 1, 0

8, 1, 15, 11, 24, 23 , 1, 0, 0

38, 17, 22, 40, 11, 39 , 0, 1, 0

26, 4, 34, 2, 21, 4 , 0, 0, 1

27, 30, 3, 22, 38, 35 , 0, 1, 0

32, 13, 22, 2, 16, 4 , 0, 1, 0

4, 16, 6, 16, 26, 19 , 0, 1, 0

4, 18, 19, 2, 36, 34 , 0, 1, 0

15, 40, 11, 20, 17, 25 , 0, 1, 0

9, 9, 40, 17, 40, 30 , 0, 1, 0

35, 4, 11, 24, 26, 16 , 0, 1, 0

10, 31, 5, 18, 13, 24 , 0, 1, 0

40, 23, 5, 12, 29, 20 , 0, 1, 0

34, 20, 23, 28, 37, 10 , 0, 1, 0

4, 17, 3, 39, 1, 30 , 0, 1, 0

7, 2, 31, 26, 19, 14 , 0, 1, 0

23, 21, 4, 16, 20, 8 , 0, 1, 0

15, 27, 29, 37, 25, 26 , 0, 1, 0

5, 35, 26, 13, 2, 23 , 0, 1, 0

40, 6, 36, 15, 40, 34 , 0, 1, 0

2, 40, 26, 16, 13, 20 , 0, 1, 0

31, 40, 8, 26, 29, 21 , 0, 0, 1

13, 12, 20, 14, 36, 4 , 0, 0, 1

18, 40, 31, 17, 19, 11 , 0, 1, 0

27, 6, 17, 22, 35, 32 , 0, 0, 1

19, 35, 9, 8, 13, 34 , 0, 1, 0

27, 40, 5, 26, 30, 10 , 0, 0, 1

19, 4, 36, 5, 22, 25 , 0, 1, 0

10, 31, 1, 5, 35, 20 , 0, 1, 0

22, 1, 11, 24, 38, 10 , 0, 0, 1

19, 20, 38, 32, 32, 2 , 1, 0, 0

26, 40, 22, 16, 16, 38 , 0, 1, 0

16, 34, 36, 4, 18, 22 , 0, 1, 0

6, 9, 8, 25, 3, 2 , 0, 1, 0

37, 16, 37, 29, 21, 22 , 0, 1, 0

32, 32, 40, 7, 18, 32 , 0, 1, 0

38, 31, 33, 33, 2, 11 , 0, 1, 0

12, 19, 33, 20, 24, 9 , 0, 1, 0

20, 25, 40, 21, 21, 8 , 1, 0, 0

3, 25, 11, 25, 14, 15 , 0, 1, 0

1, 15, 11, 3, 5, 10 , 0, 1, 0

19, 13, 17, 34, 30, 2 , 0, 0, 1

40, 23, 25, 31, 9, 18 , 0, 1, 0

15, 33, 28, 39, 29, 3 , 0, 1, 0

31, 16, 13, 16, 12, 26 , 0, 1, 0

16, 6, 1, 8, 7, 36 , 0, 1, 0

14, 23, 9, 25, 2, 6 , 0, 0, 1

12, 23, 31, 26, 1, 14 , 0, 1, 0

22, 3, 2, 18, 40, 30 , 0, 0, 1

34, 7, 38, 34, 6, 17 , 0, 1, 0

16, 34, 20, 2, 39, 23 , 0, 1, 0

35, 14, 39, 17, 30, 13 , 0, 1, 0

7, 19, 29, 24, 2, 12 , 0, 1, 0

23, 5, 18, 6, 14, 8 , 0, 1, 0

12, 3, 6, 24, 16, 32 , 0, 1, 0

10, 1, 29, 23, 24, 39 , 0, 1, 0

10, 38, 1, 5, 7, 6 , 0, 1, 0

10, 36, 36, 14, 20, 9 , 0, 1, 0

11, 30, 19, 22, 34, 39 , 0, 1, 0

36, 6, 17, 23, 14, 8 , 0, 1, 0

24, 37, 9, 34, 27, 5 , 0, 1, 0

14, 17, 13, 39, 40, 32 , 0, 1, 0

14, 4, 6, 39, 40, 17 , 0, 1, 0

16, 7, 13, 13, 39, 11 , 0, 1, 0

31, 33, 33, 4, 28, 31 , 0, 1, 0

5, 1, 32, 23, 7, 29 , 0, 1, 0

12, 23, 18, 37, 25, 24 , 0, 1, 0

6, 38, 19, 40, 30, 14 , 0, 1, 0

25, 37, 1, 39, 27, 37 , 0, 1, 0

18, 16, 5, 2, 17, 2 , 0, 1, 0

3, 33, 10, 8, 2, 3 , 0, 1, 0

24, 12, 14, 1, 10, 34 , 0, 0, 1

3, 27, 5, 31, 37, 5 , 0, 1, 0

12, 26, 9, 11, 36, 35 , 0, 1, 0

9, 8, 9, 19, 33, 34 , 1, 0, 0

32, 23, 38, 18, 7, 5 , 0, 1, 0

23, 19, 7, 40, 28, 15 , 0, 1, 0

32, 23, 28, 38, 34, 22 , 0, 1, 0

13, 39, 1, 29, 6, 11 , 0, 1, 0

11, 8, 20, 18, 34, 5 , 0, 1, 0

28, 31, 2, 30, 22, 16 , 0, 1, 0

5, 37, 17, 39, 29, 3 , 0, 1, 0

7, 1, 10, 38, 3, 34 , 0, 1, 0

3, 3, 21, 36, 2, 16 , 0, 1, 0

27, 38, 14, 36, 8, 34 , 0, 1, 0

20, 18, 26, 24, 15, 37 , 0, 1, 0

19, 24, 26, 34, 6, 28 , 0, 1, 0

23, 21, 5, 25, 20, 38 , 0, 0, 1

26, 1, 16, 22, 39, 26 , 0, 0, 1

10, 28, 12, 10, 3, 6 , 0, 1, 0

18, 14, 2, 9, 33, 32 , 0, 0, 1

34, 26, 23, 34, 19, 8 , 0, 1, 0

30, 3, 29, 22, 28, 27 , 0, 1, 0

30, 29, 22, 20, 37, 8 , 0, 0, 1

10, 4, 38, 11, 15, 16 , 0, 1, 0

30, 34, 21, 11, 23, 20 , 0, 1, 0

20, 30, 16, 28, 10, 10 , 0, 1, 0

10, 37, 14, 25, 2, 1 , 0, 1, 0

4, 24, 6, 31, 12, 23 , 0, 1, 0

11, 14, 31, 13, 5, 1 , 0, 1, 0

37, 38, 40, 20, 25, 31 , 0, 1, 0

32, 11, 1, 12, 12, 18 , 0, 1, 0

21, 32, 30, 4, 31, 30 , 0, 1, 0

5, 28, 31, 36, 7, 27 , 0, 1, 0

9, 2, 5, 14, 36, 6 , 0, 1, 0

13, 7, 39, 18, 16, 28 , 0, 1, 0

32, 30, 28, 31, 21, 36 , 0, 0, 1

16, 38, 14, 9, 7, 33 , 0, 1, 0

12, 26, 28, 24, 23, 40 , 0, 1, 0

29, 14, 38, 32, 26, 33 , 0, 1, 0

13, 24, 18, 29, 3, 22 , 0, 1, 0

9, 17, 17, 9, 1, 3 , 0, 1, 0

33, 38, 33, 16, 17, 5 , 0, 1, 0

10, 34, 29, 16, 26, 16 , 0, 1, 0

40, 14, 24, 1, 31, 21 , 0, 1, 0

31, 27, 27, 7, 5, 22 , 0, 1, 0

36, 28, 8, 24, 26, 19 , 0, 1, 0

10, 10, 19, 34, 30, 33 , 0, 1, 0

15, 24, 25, 39, 8, 14 , 0, 1, 0

9, 18, 3, 21, 20, 35 , 0, 1, 0

11, 38, 18, 18, 36, 1 , 0, 1, 0

30, 32, 7, 34, 39, 6 , 0, 1, 0

31, 8, 30, 36, 32, 34 , 0, 0, 1

14, 8, 35, 24, 30, 29 , 0, 1, 0

24, 20, 12, 22, 22, 9 , 0, 1, 0

5, 18, 13, 12, 6, 29 , 0, 1, 0

40, 21, 23, 17, 17, 15 , 0, 1, 0

22, 34, 39, 12, 39, 26 , 0, 1, 0

30, 21, 33, 39, 12, 3 , 0, 1, 0

24, 26, 40, 19, 24, 5 , 0, 0, 1

1, 9, 16, 15, 16, 24 , 0, 1, 0

35, 31, 30, 1, 18, 12 , 0, 1, 0

12, 17, 36, 3, 28, 15 , 0, 1, 0

30, 15, 34, 40, 19, 10 , 0, 1, 0

20, 23, 37, 40, 23, 34 , 0, 1, 0

17, 38, 35, 28, 14, 1 , 0, 1, 0

10, 37, 27, 21, 9, 12 , 0, 1, 0

40, 14, 16, 14, 29, 16 , 0, 1, 0

33, 28, 27, 11, 20, 31 , 0, 1, 0

16, 27, 36, 28, 31, 22 , 0, 1, 0

10, 18, 3, 34, 37, 35 , 0, 1, 0

2, 22, 24, 26, 22, 39 , 0, 1, 0

34, 12, 17, 27, 36, 39 , 0, 1, 0

21, 12, 29, 4, 12, 25 , 0, 1, 0

8, 9, 40, 28, 28, 34 , 0, 1, 0

8, 34, 9, 1, 7, 24 , 0, 1, 0

18, 29, 38, 25, 28, 39 , 0, 1, 0

7, 27, 31, 33, 34, 19 , 0, 1, 0

8, 34, 19, 14, 26, 37 , 0, 1, 0

2, 24, 15, 15, 21, 36 , 0, 1, 0

20, 28, 35, 17, 19, 2 , 0, 0, 1

1, 26, 11, 21, 3, 19 , 0, 1, 0

32, 13, 23, 27, 1, 31 , 0, 1, 0

17, 19, 32, 24, 11, 12 , 0, 1, 0

26, 28, 9, 2, 38, 26 , 0, 1, 0

24, 26, 16, 36, 24, 6 , 0, 0, 1

40, 18, 10, 13, 28, 27 , 0, 1, 0

14, 3, 13, 9, 2, 18 , 0, 0, 1

34, 17, 22, 10, 36, 17 , 0, 1, 0

33, 38, 22, 3, 32, 10 , 0, 0, 1

1, 38, 10, 27, 11, 14 , 0, 1, 0

21, 31, 39, 35, 7, 29 , 0, 1, 0

25, 37, 6, 20, 34, 15 , 0, 0, 1

6, 40, 29, 28, 24, 35 , 0, 1, 0

22, 40, 37, 39, 39, 12 , 0, 1, 0

30, 5, 15, 18, 31, 27 , 0, 1, 0

22, 23, 20, 9, 27, 39 , 1, 0, 0

1, 22, 13, 26, 24, 20 , 0, 1, 0

34, 25, 6, 3, 18, 39 , 0, 1, 0

24, 7, 19, 40, 6, 39 , 0, 0, 1

16, 29, 31, 17, 5, 24 , 0, 1, 0

15, 36, 24, 6, 26, 38 , 0, 1, 0

29, 26, 35, 6, 39, 26 , 0, 0, 1

13, 11, 11, 22, 26, 23 , 0, 1, 0

13, 39, 11, 15, 37, 40 , 0, 1, 0

13, 24, 16, 3, 36, 10 , 0, 1, 0

35, 15, 8, 2, 26, 6 , 0, 1, 0

21, 5, 35, 38, 36, 26 , 0, 1, 0

3, 36, 7, 24, 9, 10 , 0, 1, 0

26, 14, 16, 18, 26, 22 , 0, 1, 0

20, 17, 34, 14, 6, 26 , 0, 1, 0

35, 8, 36, 28, 11, 27 , 0, 1, 0

31, 18, 14, 7, 39, 33 , 0, 1, 0

3, 37, 15, 40, 22, 17 , 0, 1, 0

15, 26, 6, 33, 23, 14 , 0, 1, 0

11, 5, 23, 15, 37, 11 , 0, 1, 0

25, 6, 14, 5, 16, 10 , 0, 1, 0

21, 4, 40, 12, 25, 8 , 0, 0, 1

15, 17, 36, 18, 19, 18 , 0, 1, 0

27, 27, 2, 6, 7, 34 , 0, 1, 0

4, 20, 22, 14, 18, 40 , 0, 1, 0

18, 12, 10, 8, 14, 22 , 0, 0, 1

40, 20, 6, 35, 15, 40 , 0, 1, 0

34, 17, 36, 36, 19, 4 , 0, 1, 0

40, 34, 27, 30, 25, 5 , 0, 1, 0

26, 24, 29, 2, 36, 28 , 0, 0, 1

20, 21, 1, 5, 29, 32 , 0, 1, 0

36, 26, 9, 1, 17, 23 , 0, 1, 0

19, 16, 21, 4, 31, 36 , 0, 0, 1

9, 31, 20, 31, 17, 15 , 0, 1, 0

7, 4, 33, 25, 24, 30 , 0, 1, 0

21, 39, 19, 33, 7, 26 , 0, 0, 1

37, 22, 25, 3, 5, 5 , 0, 1, 0

34, 17, 37, 27, 30, 26 , 0, 1, 0

26, 9, 13, 13, 31, 24 , 0, 0, 1

27, 23, 37, 10, 3, 32 , 0, 0, 1

39, 35, 4, 11, 14, 9 , 0, 1, 0

36, 39, 12, 32, 39, 20 , 0, 1, 0

28, 28, 20, 17, 37, 19 , 0, 0, 1

33, 31, 31, 4, 25, 4 , 0, 1, 0

22, 8, 8, 20, 28, 3 , 0, 0, 1

16, 30, 31, 33, 39, 28 , 0, 1, 0

32, 5, 31, 38, 40, 32 , 0, 0, 1

29, 4, 30, 18, 21, 28 , 0, 1, 0

14, 13, 3, 8, 33, 7 , 0, 1, 0

1, 6, 20, 14, 5, 29 , 0, 1, 0

40, 21, 3, 11, 40, 17 , 0, 1, 0

21, 32, 26, 13, 38, 31 , 0, 1, 0

19, 15, 32, 39, 38, 8 , 0, 1, 0

39, 18, 40, 6, 32, 7 , 0, 1, 0

23, 19, 25, 13, 40, 36 , 0, 1, 0

32, 6, 9, 19, 40, 35 , 0, 1, 0

36, 27, 14, 19, 33, 39 , 0, 0, 1

18, 32, 34, 1, 33, 36 , 0, 1, 0

33, 8, 34, 24, 21, 1 , 0, 1, 0

3, 28, 37, 10, 35, 12 , 0, 1, 0

30, 2, 12, 7, 14, 9 , 0, 1, 0

35, 19, 36, 40, 1, 23 , 0, 1, 0

17, 15, 28, 17, 5, 40 , 0, 1, 0

34, 36, 31, 2, 34, 19 , 0, 1, 0

38, 17, 38, 29, 25, 7 , 0, 1, 0

37, 7, 32, 23, 1, 3 , 0, 1, 0

22, 37, 11, 10, 22, 2 , 0, 0, 1

13, 40, 32, 26, 22, 26 , 0, 1, 0

30, 25, 25, 12, 30, 24 , 0, 1, 0

3, 29, 24, 13, 28, 1 , 0, 1, 0

18, 7, 24, 29, 6, 11 , 0, 1, 0

39, 17, 19, 8, 39, 17 , 0, 1, 0

26, 35, 19, 20, 8, 33 , 0, 0, 1

18, 37, 14, 7, 7, 7 , 0, 1, 0

33, 33, 8, 1, 18, 33 , 0, 1, 0

16, 39, 4, 4, 40, 40 , 0, 1, 0

26, 30, 24, 16, 21, 27 , 0, 1, 0

20, 23, 38, 23, 35, 12 , 0, 1, 0

13, 22, 24, 21, 3, 13 , 0, 1, 0

22, 28, 31, 5, 19, 38 , 0, 0, 1

25, 37, 9, 37, 39, 33 , 0, 1, 0

9, 34, 14, 12, 36, 34 , 0, 1, 0

28, 7, 33, 31, 5, 36 , 0, 1, 0

1, 33, 9, 22, 19, 25 , 0, 1, 0

27, 20, 15, 34, 2, 20 , 0, 0, 1

36, 21, 33, 14, 10, 36 , 0, 1, 0

38, 37, 11, 25, 37, 4 , 0, 1, 0

2, 7, 33, 18, 31, 10 , 0, 1, 0

5, 11, 6, 19, 9, 19 , 0, 1, 0

34, 28, 39, 17, 17, 39 , 0, 1, 0

12, 35, 8, 31, 26, 38 , 0, 1, 0

28, 7, 29, 28, 36, 19 , 0, 1, 0

35, 14, 13, 25, 38, 31 , 0, 1, 0

7, 20, 4, 1, 11, 23 , 0, 1, 0

15, 6, 32, 8, 7, 36 , 0, 1, 0

28, 24, 24, 25, 18, 37 , 0, 0, 1

1, 38, 8, 35, 35, 15 , 0, 1, 0

1, 35, 21, 14, 1, 17 , 0, 1, 0

31, 27, 30, 29, 1, 24 , 0, 1, 0

22, 18, 26, 10, 12, 16 , 0, 0, 1

17, 2, 4, 34, 23, 18 , 0, 1, 0

1, 37, 19, 15, 34, 3 , 0, 1, 0

18, 26, 24, 8, 34, 4 , 0, 1, 0

16, 27, 1, 11, 22, 31 , 0, 1, 0

13, 35, 7, 16, 14, 35 , 0, 1, 0

7, 14, 15, 29, 11, 18 , 0, 1, 0

1, 24, 29, 31, 5, 34 , 0, 1, 0

12, 14, 20, 12, 12, 9 , 0, 1, 0

11, 29, 30, 17, 4, 38 , 0, 1, 0

19, 37, 37, 38, 8, 19 , 0, 1, 0

40, 9, 17, 12, 21, 25 , 0, 1, 0

31, 14, 9, 31, 29, 7 , 0, 1, 0

34, 33, 9, 5, 36, 28 , 0, 0, 1

22, 6, 36, 10, 29, 7 , 0, 1, 0

6, 20, 4, 20, 35, 12 , 0, 1, 0

25, 22, 39, 12, 32, 34 , 0, 1, 0

26, 37, 14, 38, 15, 16 , 0, 1, 0

12, 26, 18, 20, 31, 5 , 0, 1, 0

3, 39, 10, 1, 14, 25 , 0, 1, 0

22, 2, 27, 26, 40, 27 , 0, 1, 0

34, 30, 35, 23, 20, 14 , 0, 1, 0

3, 7, 22, 19, 38, 30 , 0, 1, 0

6, 14, 17, 1, 12, 17 , 0, 1, 0

16, 11, 37, 17, 38, 22 , 0, 1, 0

34, 1, 13, 7, 5, 34 , 0, 1, 0

26, 5, 7, 13, 34, 15 , 0, 1, 0

34, 38, 28, 38, 39, 31 , 0, 0, 1

34, 36, 29, 17, 8, 36 , 0, 1, 0

40, 36, 26, 26, 14, 28 , 0, 1, 0

7, 25, 22, 27, 15, 11 , 0, 1, 0

25, 15, 10, 20, 6, 6 , 0, 1, 0

13, 38, 9, 35, 34, 24 , 0, 1, 0

34, 34, 29, 35, 26, 28 , 0, 1, 0

8, 24, 33, 25, 1, 36 , 0, 1, 0

39, 16, 21, 38, 30, 19 , 0, 1, 0

37, 8, 8, 25, 3, 9 , 0, 1, 0

11, 19, 2, 18, 40, 30 , 0, 1, 0

28, 32, 11, 28, 12, 28 , 0, 1, 0

27, 40, 7, 8, 2, 17 , 0, 0, 1

10, 35, 33, 3, 37, 26 , 0, 1, 0

4, 2, 14, 31, 3, 11 , 0, 1, 0

12, 25, 29, 21, 14, 32 , 0, 1, 0

34, 22, 1, 21, 15, 3 , 0, 1, 0

31, 28, 25, 20, 11, 38 , 0, 0, 1

18, 11, 23, 37, 10, 8 , 0, 0, 1

15, 1, 5, 33, 35, 17 , 0, 1, 0

33, 36, 33, 28, 22, 36 , 0, 1, 0

31, 12, 39, 23, 36, 12 , 0, 1, 0

27, 9, 29, 39, 3, 10 , 0, 1, 0

15, 7, 20, 37, 37, 10 , 0, 1, 0

32, 39, 11, 8, 1, 39 , 0, 1, 0

4, 36, 5, 30, 11, 2 , 0, 1, 0

26, 17, 3, 17, 1, 14 , 0, 1, 0

21, 20, 26, 10, 8, 28 , 0, 0, 1

29, 27, 6, 32, 12, 29 , 0, 0, 1

26, 12, 20, 26, 28, 21 , 0, 1, 0

29, 31, 37, 22, 24, 23 , 0, 1, 0

30, 17, 11, 22, 39, 14 , 0, 0, 1

28, 35, 2, 10, 9, 32 , 0, 1, 0

38, 6, 36, 14, 27, 15 , 0, 1, 0

35, 26, 23, 15, 37, 35 , 0, 1, 0

17, 40, 25, 8, 19, 5 , 0, 1, 0

30, 8, 31, 32, 25, 27 , 0, 1, 0

11, 32, 19, 4, 35, 30 , 0, 1, 0

40, 22, 5, 28, 16, 16 , 0, 1, 0

36, 19, 40, 5, 15, 18 , 0, 1, 0

7, 21, 22, 23, 16, 32 , 0, 1, 0

2, 20, 7, 23, 40, 20 , 0, 1, 0

31, 34, 23, 13, 19, 37 , 0, 1, 0

38, 25, 26, 4, 36, 34 , 0, 1, 0

36, 32, 28, 6, 10, 24 , 0, 1, 0

5, 1, 9, 29, 6, 40 , 0, 1, 0

6, 34, 25, 9, 17, 13 , 0, 1, 0

6, 39, 15, 10, 38, 2 , 0, 1, 0

6, 20, 13, 18, 32, 19 , 0, 1, 0

29, 29, 23, 9, 34, 30 , 0, 1, 0

33, 6, 1, 14, 10, 27 , 0, 1, 0

25, 5, 29, 27, 37, 23 , 0, 0, 1

36, 15, 2, 8, 12, 6 , 0, 1, 0

39, 7, 25, 30, 11, 3 , 0, 1, 0

4, 25, 32, 6, 28, 31 , 0, 1, 0

34, 26, 2, 19, 30, 27 , 0, 1, 0

35, 40, 2, 25, 8, 22 , 0, 1, 0

28, 32, 20, 35, 8, 39 , 0, 1, 0

38, 5, 11, 5, 31, 20 , 0, 1, 0

20, 15, 15, 21, 4, 40 , 0, 1, 0

34, 39, 21, 29, 1, 29 , 0, 1, 0

2, 9, 11, 24, 12, 30 , 0, 1, 0

32, 15, 17, 30, 30, 1 , 0, 1, 0

39, 6, 15, 35, 39, 24 , 0, 1, 0

18, 13, 38, 37, 14, 19 , 0, 1, 0

40, 12, 29, 13, 35, 25 , 0, 1, 0

33, 17, 9, 32, 8, 4 , 0, 1, 0

40, 21, 29, 30, 32, 29 , 0, 1, 0

18, 26, 38, 7, 9, 22 , 0, 1, 0

22, 8, 22, 6, 1, 11 , 0, 1, 0

13, 38, 17, 5, 4, 14 , 0, 1, 0

22, 3, 12, 6, 26, 34 , 0, 1, 0

36, 37, 7, 17, 23, 39 , 0, 1, 0

22, 34, 24, 36, 14, 10 , 0, 1, 0

26, 2, 28, 32, 15, 25 , 0, 0, 1

32, 15, 30, 5, 32, 29 , 0, 1, 0

7, 16, 9, 4, 5, 23 , 0, 1, 0

13, 37, 13, 8, 11, 40 , 0, 1, 0

5, 3, 8, 22, 14, 34 , 0, 1, 0

1, 23, 7, 35, 28, 39 , 0, 1, 0

21, 34, 21, 30, 18, 16 , 0, 1, 0

28, 7, 33, 25, 2, 33 , 0, 0, 1

38, 4, 19, 14, 31, 15 , 0, 1, 0

Capítulo 27

Concurrencia e Hilos con Java

27.1 Datos y Ejecuciones

Contenido del archivo **Hilos.dat:**

0

30

40

50

2000

Quedan 0 unidades. El Consumidor 1 ha leido 0 unidades.

Quedan 50 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 49 unidades. El Consumidor 2 ha leido 1 unidades.

Quedan 31 unidades. El Consumidor 2 ha leido 18 unidades.

Quedan 81 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 53 unidades. El Consumidor 1 ha leido 28 unidades.

Quedan 42 unidades. El Consumidor 1 ha leido 11 unidades.

Quedan 39 unidades. El Consumidor 1 ha leido 3 unidades.

Quedan 89 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 85 unidades. El Consumidor 2 ha leido 4 unidades.

Quedan 66 unidades. El Consumidor 2 ha leido 19 unidades.

Quedan 66 unidades. El Consumidor 2 ha leido 0 unidades.

Quedan 61 unidades. El Consumidor 2 ha leido 5 unidades.

Quedan 33 unidades. El Consumidor 2 ha leido 28 unidades.

Quedan 83 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 58 unidades. El Consumidor 3 ha leido 25 unidades.

Quedan 46 unidades. El Consumidor 3 ha leido 12 unidades.

Quedan 22 unidades. El Consumidor 3 ha leido 24 unidades.

Quedan 72 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 50 unidades. El Consumidor 1 ha leido 22 unidades.

Quedan 35 unidades. El Consumidor 2 ha leido 15 unidades.

Quedan 85 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 77 unidades. El Consumidor 3 ha leido 8 unidades.

Quedan 71 unidades. El Consumidor 1 ha leido 6 unidades.

Quedan 54 unidades. El Consumidor 1 ha leido 17 unidades.

Quedan 40 unidades. El Consumidor 1 ha leido 14 unidades.

Quedan 90 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 74 unidades. El Consumidor 3 ha leido 16 unidades.

Quedan 64 unidades. El Consumidor 3 ha leido 10 unidades.

Quedan 53 unidades. El Consumidor 1 ha leido 11 unidades.

Quedan 43 unidades. El Consumidor 1 ha leido 10 unidades.

Quedan 23 unidades. El Consumidor 1 ha leido 20 unidades.

Quedan 73 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 64 unidades. El Consumidor 2 ha leido 9 unidades.

Quedan 62 unidades. El Consumidor 2 ha leido 2 unidades.

Quedan 45 unidades. El Consumidor 2 ha leido 17 unidades.

Quedan 24 unidades. El Consumidor 2 ha leido 21 unidades.

Quedan 74 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 70 unidades. El Consumidor 1 ha leido 4 unidades.

Quedan 68 unidades. El Consumidor 1 ha leido 2 unidades.

Quedan 61 unidades. El Consumidor 1 ha leido 7 unidades.

Quedan 34 unidades. El Consumidor 1 ha leido 27 unidades.

Quedan 84 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 60 unidades. El Consumidor 2 ha leido 24 unidades.

Quedan 41 unidades. El Consumidor 3 ha leido 19 unidades.

Quedan 35 unidades. El Consumidor 3 ha leido 6 unidades.

Quedan 85 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 85 unidades. El Consumidor 1 ha leido 0 unidades.

Quedan 81 unidades. El Consumidor 1 ha leido 4 unidades.

Quedan 62 unidades. El Consumidor 1 ha leido 19 unidades.

Quedan 62 unidades. El Consumidor 2 ha leido 0 unidades.

Quedan 48 unidades. El Consumidor 2 ha leido 14 unidades.

Quedan 47 unidades. El Consumidor 2 ha leido 1 unidades.

Quedan 29 unidades. El Consumidor 2 ha leido 18 unidades.

Quedan 79 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 52 unidades. El Consumidor 3 ha leido 27 unidades.

Quedan 39 unidades. El Consumidor 3 ha leido 13 unidades.

Quedan 89 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 69 unidades. El Consumidor 2 ha leido 20 unidades.

Quedan 62 unidades. El Consumidor 2 ha leido 7 unidades.

Quedan 49 unidades. El Consumidor 2 ha leido 13 unidades.

Quedan 33 unidades. El Consumidor 1 ha leido 16 unidades.

Quedan 83 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 60 unidades. El Consumidor 3 ha leido 23 unidades.

Quedan 56 unidades. El Consumidor 3 ha leido 4 unidades.

Quedan 43 unidades. El Consumidor 2 ha leido 13 unidades.

Quedan 43 unidades. El Consumidor 2 ha leido 0 unidades.

Quedan 42 unidades. El Consumidor 2 ha leido 1 unidades.

Quedan 21 unidades. El Consumidor 2 ha leido 21 unidades.

Quedan 71 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 62 unidades. El Consumidor 1 ha leido 9 unidades.

Quedan 57 unidades. El Consumidor 1 ha leido 5 unidades.

Quedan 52 unidades. El Consumidor 1 ha leido 5 unidades.

Quedan 39 unidades. El Consumidor 1 ha leido 13 unidades.

Quedan 89 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 66 unidades. El Consumidor 3 ha leido 23 unidades.

Quedan 38 unidades. El Consumidor 3 ha leido 28 unidades.

Quedan 88 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 69 unidades. El Consumidor 2 ha leido 19 unidades.

Quedan 60 unidades. El Consumidor 2 ha leido 9 unidades.

Quedan 60 unidades. El Consumidor 3 ha leido 0 unidades.

Quedan 55 unidades. El Consumidor 3 ha leido 5 unidades.

Quedan 45 unidades. El Consumidor 3 ha leido 10 unidades.

Quedan 20 unidades. El Consumidor 3 ha leido 25 unidades.

Quedan 70 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 65 unidades. El Consumidor 1 ha leido 5 unidades.

Quedan 59 unidades. El Consumidor 1 ha leido 6 unidades.

Quedan 56 unidades. El Consumidor 1 ha leido 3 unidades.

Quedan 45 unidades. El Consumidor 1 ha leido 11 unidades.

Quedan 39 unidades. El Consumidor 1 ha leido 6 unidades.

Quedan 89 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 86 unidades. El Consumidor 3 ha leido 3 unidades.

Quedan 73 unidades. El Consumidor 3 ha leido 13 unidades.

Quedan 65 unidades. El Consumidor 3 ha leido 8 unidades.

Quedan 52 unidades. El Consumidor 3 ha leido 13 unidades.

Quedan 24 unidades. El Consumidor 3 ha leido 28 unidades.

Quedan 74 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 46 unidades. El Consumidor 1 ha leido 28 unidades.

Quedan 29 unidades. El Consumidor 1 ha leido 17 unidades.

Quedan 79 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 58 unidades. El Consumidor 2 ha leido 21 unidades.

Quedan 57 unidades. El Consumidor 2 ha leido 1 unidades.

Quedan 33 unidades. El Consumidor 2 ha leido 24 unidades.

Quedan 83 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 73 unidades. El Consumidor 3 ha leido 10 unidades.

Quedan 61 unidades. El Consumidor 1 ha leido 12 unidades.

Quedan 32 unidades. El Consumidor 1 ha leido 29 unidades.

Quedan 82 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 80 unidades. El Consumidor 2 ha leido 2 unidades.

Quedan 72 unidades. El Consumidor 3 ha leido 8 unidades.

Quedan 54 unidades. El Consumidor 3 ha leido 18 unidades.

Quedan 30 unidades. El Consumidor 3 ha leido 24 unidades.

Quedan 80 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 77 unidades. El Consumidor 1 ha leido 3 unidades.

Quedan 70 unidades. El Consumidor 1 ha leido 7 unidades.

Quedan 48 unidades. El Consumidor 1 ha leido 22 unidades.

Quedan 27 unidades. El Consumidor 1 ha leido 21 unidades.

Quedan 77 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 73 unidades. El Consumidor 2 ha leido 4 unidades.

Quedan 49 unidades. El Consumidor 2 ha leido 24 unidades.

Quedan 47 unidades. El Consumidor 2 ha leido 2 unidades.

Quedan 34 unidades. El Consumidor 1 ha leido 13 unidades.

Quedan 84 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 80 unidades. El Consumidor 3 ha leido 4 unidades.

Quedan 63 unidades. El Consumidor 3 ha leido 17 unidades.

Quedan 43 unidades. El Consumidor 3 ha leido 20 unidades.

Quedan 22 unidades. El Consumidor 2 ha leido 21 unidades.

Quedan 72 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 55 unidades. El Consumidor 2 ha leido 17 unidades.

Quedan 33 unidades. El Consumidor 2 ha leido 22 unidades.

Quedan 83 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 63 unidades. El Consumidor 1 ha leido 20 unidades.

Quedan 51 unidades. El Consumidor 1 ha leido 12 unidades.

Quedan 45 unidades. El Consumidor 1 ha leido 6 unidades.

Quedan 26 unidades. El Consumidor 1 ha leido 19 unidades.

Quedan 76 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 59 unidades. El Consumidor 2 ha leido 17 unidades.

Quedan 44 unidades. El Consumidor 2 ha leido 15 unidades.

Quedan 40 unidades. El Consumidor 1 ha leido 4 unidades.

Quedan 90 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 79 unidades. El Consumidor 3 ha leido 11 unidades.

Quedan 70 unidades. El Consumidor 3 ha leido 9 unidades.

Quedan 48 unidades. El Consumidor 3 ha leido 22 unidades.

Quedan 43 unidades. El Consumidor 3 ha leido 5 unidades.

Quedan 16 unidades. El Consumidor 3 ha leido 27 unidades.

Quedan 66 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 60 unidades. El Consumidor 2 ha leido 6 unidades.

Quedan 42 unidades. El Consumidor 2 ha leido 18 unidades.

Quedan 23 unidades. El Consumidor 2 ha leido 19 unidades.

Quedan 73 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 45 unidades. El Consumidor 3 ha leido 28 unidades.

Quedan 38 unidades. El Consumidor 3 ha leido 7 unidades.

Quedan 88 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 71 unidades. El Consumidor 2 ha leido 17 unidades.

Quedan 71 unidades. El Consumidor 2 ha leido 0 unidades.

Quedan 47 unidades. El Consumidor 2 ha leido 24 unidades.

Quedan 24 unidades. El Consumidor 3 ha leido 23 unidades.

Quedan 74 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 69 unidades. El Consumidor 3 ha leido 5 unidades.

Quedan 42 unidades. El Consumidor 3 ha leido 27 unidades.

Quedan 13 unidades. El Consumidor 3 ha leido 29 unidades.

Quedan 63 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 58 unidades. El Consumidor 2 ha leido 5 unidades.

Quedan 46 unidades. El Consumidor 2 ha leido 12 unidades.

Quedan 46 unidades. El Consumidor 1 ha leido 0 unidades.

Quedan 21 unidades. El Consumidor 1 ha leido 25 unidades.

Quedan 71 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 64 unidades. El Consumidor 3 ha leido 7 unidades.

Quedan 40 unidades. El Consumidor 2 ha leido 24 unidades.

Quedan 90 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 89 unidades. El Consumidor 1 ha leido 1 unidades.

Quedan 68 unidades. El Consumidor 1 ha leido 21 unidades.

Quedan 45 unidades. El Consumidor 1 ha leido 23 unidades.

Quedan 18 unidades. El Consumidor 3 ha leido 27 unidades.

Quedan 68 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 65 unidades. El Consumidor 2 ha leido 3 unidades.

Quedan 49 unidades. El Consumidor 2 ha leido 16 unidades.

Quedan 24 unidades. El Consumidor 2 ha leido 25 unidades.

Quedan 74 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 59 unidades. El Consumidor 3 ha leido 15 unidades.

Quedan 44 unidades. El Consumidor 3 ha leido 15 unidades.

Quedan 32 unidades. El Consumidor 3 ha leido 12 unidades.

Quedan 82 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 77 unidades. El Consumidor 1 ha leido 5 unidades.

Quedan 62 unidades. El Consumidor 1 ha leido 15 unidades.

Quedan 44 unidades. El Consumidor 1 ha leido 18 unidades.

Quedan 37 unidades. El Consumidor 1 ha leido 7 unidades.

Quedan 87 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 64 unidades. El Consumidor 3 ha leido 23 unidades.

Quedan 47 unidades. El Consumidor 3 ha leido 17 unidades.

Quedan 44 unidades. El Consumidor 2 ha leido 3 unidades.

Quedan 43 unidades. El Consumidor 2 ha leido 1 unidades.

Quedan 18 unidades. El Consumidor 2 ha leido 25 unidades.

Quedan 68 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 41 unidades. El Consumidor 1 ha leido 27 unidades.

Quedan 24 unidades. El Consumidor 1 ha leido 17 unidades.

Quedan 74 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 51 unidades. El Consumidor 3 ha leido 23 unidades.

Quedan 42 unidades. El Consumidor 1 ha leido 9 unidades.

Quedan 16 unidades. El Consumidor 1 ha leido 26 unidades.

Quedan 66 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 50 unidades. El Consumidor 2 ha leido 16 unidades.

Quedan 28 unidades. El Consumidor 2 ha leido 22 unidades.

Quedan 78 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 74 unidades. El Consumidor 1 ha leido 4 unidades.

Quedan 50 unidades. El Consumidor 3 ha leido 24 unidades.

Quedan 35 unidades. El Consumidor 3 ha leido 15 unidades.

Quedan 85 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 74 unidades. El Consumidor 2 ha leido 11 unidades.

Quedan 59 unidades. El Consumidor 2 ha leido 15 unidades.

Quedan 59 unidades. El Consumidor 1 ha leido 0 unidades.

Quedan 30 unidades. El Consumidor 1 ha leido 29 unidades.

Quedan 80 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 53 unidades. El Consumidor 3 ha leido 27 unidades.

Quedan 43 unidades. El Consumidor 3 ha leido 10 unidades.

Quedan 19 unidades. El Consumidor 3 ha leido 24 unidades.

Quedan 69 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 69 unidades. El Consumidor 1 ha leido 0 unidades.

Quedan 45 unidades. El Consumidor 1 ha leido 24 unidades.

Quedan 32 unidades. El Consumidor 1 ha leido 13 unidades.

Quedan 82 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 78 unidades. El Consumidor 2 ha leido 4 unidades.

Quedan 52 unidades. El Consumidor 2 ha leido 26 unidades.

Quedan 32 unidades. El Consumidor 2 ha leido 20 unidades.

Quedan 82 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 74 unidades. El Consumidor 1 ha leido 8 unidades.

Quedan 74 unidades. El Consumidor 1 ha leido 0 unidades.

Quedan 65 unidades. El Consumidor 1 ha leido 9 unidades.

Quedan 56 unidades. El Consumidor 1 ha leido 9 unidades.

Quedan 52 unidades. El Consumidor 3 ha leido 4 unidades.

Quedan 44 unidades. El Consumidor 3 ha leido 8 unidades.

Quedan 18 unidades. El Consumidor 2 ha leido 26 unidades.

Quedan 68 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 48 unidades. El Consumidor 2 ha leido 20 unidades.

Quedan 42 unidades. El Consumidor 2 ha leido 6 unidades.

Quedan 27 unidades. El Consumidor 1 ha leido 15 unidades.

Quedan 77 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 77 unidades. El Consumidor 1 ha leido 0 unidades.

Quedan 62 unidades. El Consumidor 1 ha leido 15 unidades.

Quedan 59 unidades. El Consumidor 1 ha leido 3 unidades.

Quedan 55 unidades. El Consumidor 1 ha leido 4 unidades.

Quedan 31 unidades. El Consumidor 1 ha leido 24 unidades.

Quedan 81 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 71 unidades. El Consumidor 3 ha leido 10 unidades.

Quedan 49 unidades. El Consumidor 3 ha leido 22 unidades.

Quedan 24 unidades. El Consumidor 3 ha leido 25 unidades.

Quedan 74 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 46 unidades. El Consumidor 3 ha leido 28 unidades.

Quedan 22 unidades. El Consumidor 3 ha leido 24 unidades.

Quedan 72 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 43 unidades. El Consumidor 2 ha leido 29 unidades.

Quedan 27 unidades. El Consumidor 2 ha leido 16 unidades.

Quedan 77 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 68 unidades. El Consumidor 3 ha leido 9 unidades.

Quedan 43 unidades. El Consumidor 3 ha leido 25 unidades.

Quedan 25 unidades. El Consumidor 3 ha leido 18 unidades.

Quedan 75 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 73 unidades. El Consumidor 2 ha leido 2 unidades.

Quedan 61 unidades. El Consumidor 2 ha leido 12 unidades.

Quedan 39 unidades. El Consumidor 2 ha leido 22 unidades.

Quedan 89 unidades. El Productor 2 ha grabado 50 unidades.

Quedan 85 unidades. El Consumidor 3 ha leido 4 unidades.

Quedan 68 unidades. El Consumidor 3 ha leido 17 unidades.

Quedan 57 unidades. El Consumidor 1 ha leido 11 unidades.

Quedan 41 unidades. El Consumidor 1 ha leido 16 unidades.

Quedan 25 unidades. El Consumidor 1 ha leido 16 unidades.

Quedan 75 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 67 unidades. El Consumidor 2 ha leido 8 unidades.

Quedan 67 unidades. El Consumidor 3 ha leido 0 unidades.

Quedan 65 unidades. El Consumidor 3 ha leido 2 unidades.

Quedan 46 unidades. El Consumidor 3 ha leido 19 unidades.

Quedan 36 unidades. El Consumidor 3 ha leido 10 unidades.

Quedan 86 unidades. El Productor 1 ha grabado 50 unidades.

Quedan 82 unidades. El Consumidor 2 ha leido 4 unidades.

Quedan 64 unidades. El Consumidor 2 ha leido 18 unidades.

Quedan 48 unidades. El Consumidor 2 ha leido 16 unidades.

Quedan 46 unidades. El Consumidor 2 ha leido 2 unidades.

Quedan 42 unidades. El Consumidor 2 ha leido 4 unidades.

Quedan 19 unidades. El Consumidor 2 ha leido 23 unidades.

Fin de la ejecución.

Capítulo 28

Anomalía de Belady con Matlab

28.1 Datos y Ejecuciones

Contenido del archivo **Anomalia.dat:**

Vector de requerimientos de páginas para la simulación:

622 684 182 511 999 231 233 655 611 501 3 281 558 359 421 446 12 985 980 847 11 866 957 967 224 990 516 267 982 487 924 629 308 803 134 573 175 802 399 982 871 11 492 676 578 17 453 985 781 1000 389 972 228 770 221 956 498 941 155 457 842 976 30 112 409 700 547 68 886 737 949 985 621 500 623 703 765 891 577 633 535 285 360 178 365 848 543 264 497 447

Pares celdas de página disponibles y fallos de páginas ocurridos:

8 90

10 90

12 89

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

471 590 792 692 95 212 714 16 192 330 16 255 483 891 675 736 874 893 584 552 3 811 238 435 584 240 629 792 506 334 401 419 899 333 801 875 745 932 209 533 278 199 803 788 178 353 668 61 174 499 771 74 944 280 601 843 120 492 344 645 570 49 648 148 390 856 204 604 10 975 447 274 138 932 53 104 588 470 16 879 449 797 104 631 742 553 405 16 753 664

Pares celdas de página disponibles y fallos de páginas ocurridos:

8 86

10 86

12 86

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

444 876 979 864 979 26 827 777 286

Pares celdas de página disponibles y fallos de páginas ocurridos:

2 8

3 8

4 8

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

553 865 706 923 273 591 822 483 706

Pares celdas de página disponibles y fallos de páginas ocurridos:

2 9

3 9

4 9

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

223 774 721 174 115 506 678 53 549 627 667 333 735 375 1 684 179 443 974 772 325 115 599 726 781 586 799 99 934 867 930 448 935 651 633 726 130 444 353 909 869 791 316 340 318 322 586 39 653 171 895 169 669 593 76 225 539 823 161 326 952 110 591 951 87 923 602 356 134 122 693 12 533 547 0 653 160 981 182 339 376 534 448 915 436 886 543 203 666 331 961 733 713 551 5 34 78 43 990 703 553 647 364 236 846 90 776 762 499 471 600 889 748 184 96 60 288 718 155 113 211 396 278 651 457 47 504 377 923 529 244 717 598 784 129 980 817 394 398 927 943 577 679 784 398 998 604 968 945 990 9 581 995 660 381 23 519 393 335 463 395 36 878 994 98 117 173 412 784 345 210 447 649 294 6 616 164 865 826 498 14 543 55 603 678 45 187 64 620 665 519 864 281 283 881 826 190 902 9 57 965 918 413 397 432 994 181 771 553 179 864 141 63 648 778 328 478 178 741 556 781 720 955 385 786 192 258 73 886 546 398 58 516 113 955 757 817 366 118 584 415 637 648 290 452 163 552 64 415 82 263 859 2 451 88 520 680 203 716 607 560 40 208 614 842 244 583 356 674 153 883 384 620 703 972 636 34 856 617 149 715 444 601 200 437 349 715 282 323 909 415 974 280 843 348 178 522 867 720 343 817 149 596 402 373 978 158 499 749 347 521 914 349 54 510 446 161 212 290 806 51 746 596 354 250 704 226 293 331 904 987 328 556 898 812 413 686 548 615 9 658 956 437 750 250 109 490 109 443 494 16 172 755 307 23 844 238 418 214 141 889 974 68 13 360 833 511 246 583 809 758 71 90 825 893 735 279 606 204 631 311 596 362 934 348 706 629 209 74 985 347 651 7 318 891 855 32 761 361 4 362 586 842 538 941 880 122 744 548 864 278 948 126 116 513 976 66 49

609 459 364 605 368 808 787 651 965 810 627 851 227 243 279 969 269 388 371 585 674 767 859 370 570 372 929 69 448 923 375 874 714 703 418 606 94 500 678 12 728 521 289 42 890 276 49 238 171 117 641 847 806 519 224 91 81 51 694 553 735 69 719 278 763 888 603 451 777 980 246 876 390 12 197 638 31 28 364 628 547 899 791 168 118 192 746 731 27 479 343 713 509 671 85 264 491 616 82 620 556 638 477 185 561 52 924 841 342 550 283 459 683 668 887 924 855 254 983 411 49 712 162 515 801 241 890 980 968 218 482 592 868 472 592 705 256 68 780 672 462 990 347 286 465 870 454 153 732 663 693 615 622 976 250 125 40 620 578 819 887 901 718 467 279 745 724 291 844 85 515 805 636 384 605 834 958 170 832 248 509 479 94 22 167 685 981 968 51 485 87 219 635 160 727 765 171 887 343 173 477 990 792 318 987 882 266 863 708 852 515 288 980 674 465 133 609 770 70 832 850 679 551 896 715 783 618 298 281 434 264 952 445 89 882 852 859 747 619 986 361 198 531 652 219 463 381 896 972 645 456 599 420 96 665 560 393 331 753 549 175 788 734 490 717 696 520 986 651 269 988 466 990 999 41 371 69 732 217 334 118 176 698 187 847 953 141 918 597 888 679 507 867 221 419 895 210 272 965 602 874 818 77 53 513 524 521 150 36 430 988 627 26 412 342 206 657 945 994 575 643 457 51 819 956 408 20 848 183 962 223 218 627 952 842 753 950 658 570 231 428 994 840 825 881 579 330 333 796 231 646 201 993 301 236 378 29 826 798 15 317 693 113 318 607 568 985 313 930 477 132 946 719 874 586 154 856 136 450 348 147 18 128 235 721 739 202 371 32 244 290 713 362 401 507 102 767 58 98 178 176 537 971 818 907 696 228 425 560 169 623 847 710 845 157 415 583 701 272 715 880 714 469 663 756 74 599 185 175 54 526 831 252 794 108 698 383 873 255 60 26 61 371 428 422 469 584 207 248 910 884 215 811 601 923 356 17 878 488 437 273 405 135 963 327 202 262 88 512 996 408 777 568 938 831 309 694 171 729 936 82 657

Pares celdas de página disponibles y fallos de páginas ocurridos:

240 707

300 669

360 640

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

194 820 451 580 707 594 281 802 152 811 930 594 524 36 855 367 408 496 41 171 672 555 670 884 768 254 697 309 325 452 269 889 517 125 78 160 314 619 918 177 973 772 783 168 12 998 114 519 896 554 179 353 990 88 714 768 508 222 495 104 483 818 94 762 59 288 696 963 757 776 726 271 331 295 986 309 624 61 63 723 183 788 710 707 384 989 460 839 172 19 28 443 255 646 348 905 172 780 260 810 27 287 671 516 883 292 24 234 872 522 628 927 300 900 652 301 170 888 553 966 204 347 805 474 598 708 650 906 999 847 577 771 837 784 400 938 769 633 102 429 982 603 29 707 849 526 824 851 333 650 752 180 891 257 983 813 579 882 965 553 459 674 646 445 421 990 87 434 313 898 677 418 990 528 769 135 309 384 631 263 469 872 479 71 837 960 536 330 679 580 135 773 951 590 348 954 988 511 562 255 581 602 2 399 370 707 532 99 533 200 703 632 9 815 655 842 720 171 171 919 992 73 921 632 447 279 399 180

298 578 760 348 745 696 578 949 822 8 135 180 11 143 536 809 45 243 803 446 527 751 960 751 140 18 720 986 570 989 57 470 761 447 53 827 277 581 736 380 63 239 832 59 36 352 767 472 42 538 754 578 91 274 548 936 756 948 965 732 490 236 606 20 615 888 979 208 368 644 637 46 380 318 136 416 528 152 717 935 383 136 101 677 763 128 394 971 633 734 611 584 461 240 957 684 981 348 529 222 140 21 70 311 222 458 363 703 427 32 461 527 901 750 171 712 970 854 549 333 26 512 904 857 67 592 366 451 488 264 158 27 940 144 207 921 295 831 888 745 749 371 966 854 923 158 209 36 833 815 973 841 489 199 84 927 590 931 856 895 858 26 152 631 27 451 370 472 771 402 999 38 908 777 896 711 137 964 365 438 689 938 135 161 236 355 763 621 288 339 881 340 312 572 400 780 546 933 710 222 643 59 473 253 606 548 812 360 969 259 530 648 405 72 995 539 482 559 755 930 461 31 561 220 189 300 533 142 615 58 447 258 109 174 339 439 215 192 900 484 504 301 130 541 221 402 680 958 968 178 740 470 867 607 616 204 510 975 306 489 152 892 719 335 32 43 811 451 275 991 432 779 577 474 978 325 137 434 17 979 975 120 437 877 988 773 642 903 524 946 570 152 340 354 867 214 693 832 69 549 666 47 919 544 18 743 525 945 565 559 256 962 661 649 876 596 32 989 369 241 265 889 455 615 75 223 10 785 196 946 727 391 201 988 658 596 428 786 360 758 445 112 245 185 832 149 662 652 430 899 185 86 37 619 114 671 932 361 523 952 668 171 183 423 992 625 45 394 753 307 618 126 875 796 810 407 162 757 166 935 395 978 340 93 508 98 941 941 689 61 568 901 240 784 656 749 799 756 967 590 317 992 552 422 474 257 291 811 774 94 525 234 686 805 309 125 852 233 161 255 573 289 297 827 145 878 323 517 106 943 529 810 536 446 316 797 655 761 652 145 453 952 549 53 409 853 746 330 733 359 688 835 242 587 499 43 420 288 198 39 684 378 699 91 522 290 745 310 522 86 269 482 509 853 49 555 260 574 318 430 455 512 170 264 714 868 341 787 539 707 109 89 731 547 985 499 730 813 235 301 379 389 493 45 467 841 987 973 342 812 68 661 174 468 925 551 783 779 41 383 395 607 946 707 125 287 7 18 408 142 353 584 700 191 104 468 379 316 130 417 554 187 671 879 43 125 465 183 363 766 415 370 157 104 814 538 292 467 698 640 758 370 748 885 903 51 666 640 146 66 705 428 764 799 514 183 96 618 806 247 674 161 570 957 52 355 621 820 969 846 743 441 776 700 976 353 545 485 733 318 295 370 805 832 826 368 295 732 459 73 89 241 329 716 797 589 601 842 32 4 88 796 29 348 25 822 843 268 25 78 915 553 634 138 417 531 759 147 978 394 566 517 531 470 73 711 947 297 371 136 378 592 712 510 849 28 488 654 46 479 856 727 989 561 749 484 951 67 685 813 276 475

Pares celdas de página disponibles y fallos de páginas ocurridos:

240 702

300 666

360 636

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

415 703 646 137 984 445 705 811 119 593 673 197 536 367 743 438 148 870 417 171 277 901 194 135 414 21 510 457 92 497 974 659 573 887 920 863 398 310 149 199 706 810

479 59 842 770 495 985 321 370 432 258 245 140 83 769 299 670 743 35 313 326 236 308 273 273 120 567 19 428 136 739 882 76 119 344 223 88 81 609 706 813 745 963 451 406 37 949 31 52 514 767 860 653 549 43 216 75 85 293 620 264 417 830 298 920 310 434 257 992 467 102 418 601 518 188 768 68 715 936 604 314 387 640 289 110 104 182 912 716 216 148 89 519 643 115 938 827 182 304 547 321 840 54 591 409 689 891 526 409 218 138 115 117 215 280 482 379 704 342 587 325 717 639 125 345 398 207 647 146 992 369 347 653 335 753 106 295 153 65 509 231 863 902 995 438 607 639 329 710 579 896 130 807 217 685 617 519 769 419 910 173 76 242 8 982 126 820 266 368 573 81 336 752 431 691 486 740 936 253 190 403 350 342 215 621 431 33 147 592 92 117 799 537 198 532 579 828 168 348 233 940 257 340 101 838 370 645 323 401 952 73 941 63 694 634 612 770 464 110 762 296 516 292 616 821 710 920 183 382 733 526 306 848 808 720 323 720 161 72 727 792 137 638 187 619 169 397 203 263 565 946 133 618 170 772 738 65 701 699 269 585 614 29 146 722 226 799 637 938 949 194 788 385 222 531 918 299 693 383 313 34 533 161 186 55 277 687 397 587 724 349 65 735 246 535 661 73 450 745 788 606 6 755 855 329 687 502 116 971 202 540 80 325 438 469 541 743 743 30 620 724 838 454 66 518 692 418 84 935 868 922 400 28 568 229 14 227 423 471 3 889 271 929 727 55 208 360 352 331 990 326 325 93 518 576 830 575 432 620 864 108 158 641 133 509 200 425 405 519 829 473 875 680 709 22 83 362 996 360 962 92 67 256 853 831 461 719 472 195 111 88 635 244 502 923 538 101 184 347 516 804 848 84 533 992 397 588 235 565 467 87 491 122 509 551 289 807 844 26 701 278 10 200 753 124 935 84 218 631 911 917 161 69 655 193 535 221 288 182 987 802 546 651 781 613 182 599 891 734 127 826 642 600 941 476 246 691 717 47 777 808 621 400 112 659 994 48 535 881 17 823 673 205 134 539 98 176 835 870 466 110 515 215 618 333 124 234 972 889 974 331 729 628 2 617 891 570 864 118 761 19 622 947 99 640 159 782 577 119 265 87 28 662 14 555 16 451 294 338 612 752 116 642 218 3 668 839 735 103 823 775 797 111 392 197 200 948 676 731 818 433 473 204 803 813 310 197 437 811 449 621 905 453 795 932 696 881 724 259 685 293 338 731 565 666 294 342 695 824 456 599 208 783 686 93 947 262 463 144 743 408 667 987 419 60 306 514 170 192 352 904 560 210 657 165 690 666 134 445 915 826 667 618 209 969 930 637 123 311 177 764 397 802 113 787 849 863 253 899 857 660 932 705 862 508 653 894 221 336 497 667 118 907 27 888 584 580 862 744 331 511 174 822 136 976 206 89 616 522 941 418 199 420 456 340 60 102 41 120 671 732 711 608 996 449 889 342 276 53 655 943 129 226 766 772 388 874 562 579 242 324 465 441 172 241 336 607 806 155 206 550 117 944 677 220 627 592 269 801 577 847 929 198 908 880 417 499 235 500 898 459 1 166 454 507 982 342 335 8 596 175 605 211 275 530 646 396 327 668 846 991 289 961 164 830 657 46 542 61 1 887 883 895 667 760 777 340 437 148 487 300 864 296 506 568 986 752 162 475 24 368 764 857 723 95 270 569 405 131 625 706 821 39 556 78 215 709 204 132 191 318 336 625 839 507 4 606 695 788 609 851 608 648 540 696 753 652 217 383 143 87 677 680 407 773 195 539 994 576 808 963 635 611 445 77 165 457 107 503 400 948 469 84 163 193 562 620 323 131 128 299 844 698 28 28 770 571 863 491 138 957 191 419 311 160 838 677 344 206 490 422 617 550 378 892 924 514 245 353

Pares celdas de página disponibles y fallos de páginas ocurridos:

240 711

300 666

360 646

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

563 623 142 314 991 869 640 44 12 846 588 14 985 593 253 130 678 19 443 843 62 355 266 416 375 529 267 160 284 927 363 581 955 445 128 587 449 52 374 795 500 341 910 945 813 133 394 490 141 260 420 295 715 331 850 259 330 241 11 218 876 196 668 343 90 876 942 266 255 622 619 699 355 633 875 370 782 252 115 352 57 535 536 476 727 219 311 155 732 514 27 751 144 771 189 859 150 340 600 456 190 794 976 893 454 814 284 620 396 358 376 530 689 7 749 249 228 744 500 185 538 569 167 78 612 875 883 83 894 60 490 152 98 915 169 365 353 792 510 248 956 3 97 52 237 237 929 199 140 69 850 395 588 213 459 143 937 329 387 498 496 602 28 140 167 581 36 736 595 602 986 859 227 339 579 964 734 354 232 484 532 364 854 578 56 462 752 1 349 243 130 898 889 86 14 291 614 44 440 598 141 923 388 496 410 635 24 401 59 793 91 990 96 350 583 842 367 72 124 190 560 224 413 227 320 481 657 499 39 390 670 227 551 784 74 370 663 770 46 39 858 824 731 835 483 477 166 302 782 107 360 788 731 658 97 193 578 249 577 195 608 221 5 490 747 199 885 721 585 88 205 750 899 987 693 12 316 965 607 715 337 175 24 981 517 26 816 873 108 829 111 820 937 606 598 95 871 71 786 61 24 216 314 284 108 791 783 42 306 237 356 319 325 373 728 822 353 519 346 807 354 929 38 965 785 946 892 645 864 772 790 8 855 765 6 817 717 8 128 799 99 660 465 185 268 831 104 745 482 721 4 828 951 552 681 292 304 328 523 828 625 664 631 109 631 212 914 73 994 900 794 459 992 649 430 972 973 544 257 311 991 453 440 554 481 113 502 14 275 134 349 457 797 572 801 407 79 689 462 739 700 294 903 102 847 721 231 512 670 993 858 820 470 79 160 860 341 186 885 171 204 987 165 839 376 792 146 829 64 658 141 824 435 444 295 935 602 875 951 292 294 18 509 570 80 284 878 824 999 751 908 124 34 460 168 292 700 632 751 713 74 932 92 598 718 415 839 369 245 850 891 652 204 584 322 332 320 130 246 761 395 176 238 741 130 208 43 179 743 651 299 90 653 554 304 736 908 714 866 424 950 342 36 277 296 330 960 904 248 101 997 81 318 681 758 436 295 31 656 184 873 854 195 565 287 911 178 903 120 116 477 985 535 51 775 491 293 900 908 182 948 924 890 699 770 390 659 138 49 871 783 583 731 580 743 468 545 63 592 929 583 704 42 685 457 34 468 120 72 321 310 855 304 828 859 552 932 859 25 941 111 888 247 757 243 116 684 692 821 276 425 967 249 930 361 542 150 645 871 848 650 783 22 872 416 859 596 376 783 790 446 441 894 226 923 110 104 383 925 167 363 589 604 20 771 26 797 627 120 383 666 966 196 435 278 73 3 3 120 962 346 370 439 366 751 68 341 791 918 458 581 945 4 154 521 268 784 24 202 492 454 597 587 888 672 458 491 967 673 178 295 564 680 179 163 594 804 696 309 746 643 7 999 484 622 288 499 504 458 323 678 681 169 114 955 351 164 430 915 414 448 501 225 591 364 290 491 395 458 663 929 813 178 759 617 85 210 774 358 136 144 886 515 616 288 252 401 488 327 570 584 703 83 172 357 473 621 881 356 988 860 960 554 437 795 422 913 271 556 851 555 573 663 311 368 783 828 307 796 535 318 599 413 701 320 247 582 264 859 781 531 43 60 749 283 363 887 687 500 241 524 337 4 585 766 148 441 349 708 973 365 690 918 590 284 714 676 917 227 159 563 123 175 274 160 915 198 722 325 888

69 736 458 623 686 571 953 758 732 1 775 784 76 489 155 798 425 404 968 460 230 375 350 199 275 56 584 540 636 191 368 120 829 307 678 804 989 599 300 671 738 121 708 661 272 159 936 919 759 768 484 820 941 52 882 957 128 465 398 110 965 708 741 867 763 690 645 591 260 707 827 271 682 378 989 967 641 957 413 621 894 451 966 685 474 457 902 911 540 688

Pares celdas de página disponibles y fallos de páginas ocurridos:

240 703

300 679

360 651

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

830 266 128 559 353 864 783 456 850 860 604 821 240 11 570 992 541 752 413 918 748 48 515 924 188 474 420 659 641 133 84 930 276 276 39 280 425 663 883 279 921 7 600 454 151 809 387 682 888 825 267 670 822 883 128 469 707 573 305 65 208 245 895 696 265 429 647 126 647 916 786 961 766 585 472 656 671 521 727 740 75 521 111 528 70 564 167 762 48 562 170 82 779 142 314 902 704 54 117 300 973 914 973 47 437 805 870 207 833 975 924 543 27 799 158 414 780 916 289 366 447 114 524 859 139 448 213 182 615 261 306 848 621 582 543 403 82 535 422 314 711 155 177 93 680 733 870 16 698 509 361 823 183 612 302 706 569 126 156 787 950 475 425 917 24 853 661 566 907 661 679 581 383 919 276 967 74 308 318 336 973 296 677 293 808 826 944 233 763 7 641 546 698 841 656 278 759 146 671 596 505 885 568 890 249 136 617 28 218 209 88 135 116 498 511 418 947 856 340 9 867 983 604 14 747 147 411 952 601 828 891 893 388 52 842 45 80 7 568 527 19 47 184 466 504 496 479 243 283 819 69 73 780 212 79 650 939 656 64 706 18 799 773 937 748 377 168 117 374 87 78 849 684 765 572 54 569 84 86 76 465 835 89 464 492 14 728 748 627 151 891 828 135 625 129 611 802 845 518 256 790 534 243 741 611 309 80 568 814 196 943 188 286 388 642 183 24 724 939 977 818 518 62 802 489 501 998 114 742 305 711 863 135 62 538 701 230 485 665 429 636 381 299 472 780 347 222 673 361 826 537 393 925 514 143 751 336 324 490 714 226 266 275 346 177 394 61 817 993 642 621 571 8 640 205 150 655 766 46 556 856 166 278 78 75 742 792 713 750 836 86 29 982 808 415 930 435 741 413 44 68 181 103 403 169 125 479 783 373 610 96 649 743 509 565 98 200 715 858 655 920 768 385 714 228 694 970 325 761 561 241 799 842 111 691 549 599 9 411 92 445 853 617 173 387 184 875 911 951 637 760 29 398 471 379 747 613 279 121 620 836 333 796 51 592 425 926 776 27 722 671 410 226 455 155 531 329 721 195 421 878 229 880 805 523 540 79 266 749 636 120 357 900 790 773 678 606 930 988 96 395 923 876 334 523 122 384 570 600 433 89 922 839 789 420 340 580 101 80 86 524 411 534 124 153 166 86 602 675 44 490 644 10 578 961 974 536 534 414 292 33 162 324 675 820 424 710 32 132 794 288 156 675 763 932 501 476 253 926 794 50 383 181 822 309 785 450 355 878 114 193 441 146 721 814 588 568 861 654 32 153 657 328 695 724 810 226 835 898 66 870 343 748 878 934 468 422 815 908 484 819 879 789 906 27 383 197 752 890 956 621 691 942 969 456 407 391 243 502 106 983 419 173 906 351 383

363 429 655 636 945 66 984 113 715 899 173 820 169 435 802 311 638 76 40 377 38 46 682 637 453 442 362 246 129 933 646 790 9 302 636 211 586 206 664 755 565 940 132 99 392 468 408 451 130 387 886 316 352 154 735 239 37 664 452 572 597 393 82 582 113 879 681 177 150 71 985 390 790 502 815 171 913 333 693 628 13 11 768 860 981 252 889 528 29 461 234 171 173 454 473 662 990 320 693 584 696 68 779 908 339 927 289 590 25 992 316 502 150 793 484 412 28 856 235 311 253 832 74 142 362 490 434 116 485 867 649 979 633 35 315 573 995 683 960 6 123 784 932 313 949 505 977 839 317 825 382 7 525 739 555 864 704 969 760 127 210 109 887 154 903 5 494 950 726 156 890 789 362 112 406 524 9 587 647 77 797 843 59 464 281 749 933 236 178 795 341 245 577 278 850 476 882 426 826 407 189 396 384 333 114 306 552 876 667 416 80 78 836 583 671 602 406 530 531 739 526 223 467 630 158 953 857 916 866 763 58 95 388 147 906 635 589 8 627 486 176 802 731 900 239 394 895 277 659 726 874 633 241 565 352 619 319 378 304 617 458 936 897 511 743 156 610 394 29 890 912 960 186 784

Pares celdas de página disponibles y fallos de páginas ocurridos:

240 703

300 674

360 646

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

526 304 594 255 721 775 610 652 144 499 762 754 362 864 863 626 743 625 110 253 322 345 660 84 996 67 435 815 372 919 667 695 936 392 423 665 338 371 242 78 579 612 729 979 142 86 92 23 630 621 860 19 766 805 939 715 781 547 22 348 937 195 343 213 777 834 671 806 680 292 202 426 378 123 487 986 63 702 113 702 797 28 769 753 968 347 972 911 302 67 846 108 15 311 130 860 488 713 739 787 41 686 877 88 93 178 102 644 219 909 588 647 143 426 802 764 402 813 437 214 333 281 803 263 780 923 265 912 566 737 658 881 619 911 696 565 34 579 246 49 540 782 898 276 511 733 531 600 709 65 401 828 47 55 534 939 81 392 288 549 831 610 377 578 148 620 922 12 169 565 884 334 756 907 33 12 928 80 784 980 313 749 565 89 197 126 58 702 315 112 411 363 241 114 863 866 578 145 538 819 532 328 993 343 735 188 236 975 72 894 925 991 829 948 476 758 384 15 127 837 51 284 664 372 10 420 94 693 190 415 173 319 227 246 501 586 733 457 388 63 917 232 481 378 72 620 174 993 829 658 395 289 614 678 266 836 990 96 537 517 272 581 482 111 866 341 159 972 346 711 984 64 530 75 694 928 326 512 419 318 354 407 30 996 83 179 899 143 335 16 23 220 963 513 184 144 887 792 529 895 172 700 939 458 559 39 200 533 647 627 527 638 760 651 257 386 848 53 292 323 989 940 676 851 760 369 791 454 24 141 829 63 5 224 245 36 388 697 150 765 647 315 481 409 904 147 536 213 368 252 870 318 25 820 402 341 756 224 475 196 190 423 12 842 284 499 538 434 402 906 893 847 927 895 3 673 934 158 52 867 197 33 911 100 462 485 927 235 387 848 972 90 612 860 125 243 771 632 939 124 938 282 712 978 597 996 8 320 370 340 250 738 182 109 154 997 765 147 112 359 291 906 732 103 774 852 894 250 771 129 471 181 434 299 764 242 24 289

557 693 581 221 135 953 927 533 496 757 72 609 617 221 195 426 282 533 111 133 919 18 276 40 334 693 623 130 546 988 478 593 496 849 627 129 225 291 185 983 329 929 726 806 571 113 117 257 874 164 617 709 126 328 580 97 780 873 631 607 642 399 817 260 456 681 482 566 397 302 781 401 194 189 967 181 572 348 611 52 948 339 941 844 958 916 300 659 13 558 730 516 488 132 821 195 391 895 7 324 855 695 733 491 651 982 530 997 205 465 991 77 519 480 985 446 187 106 970 67 104 37 945 162 580 724 33 996 154 24 381 919 723 774 851 659 573 226 9 268 315 776 350 239 642 615 454 17 775 405 301 73 970 97 707 427 954 253 195 507 939 518 160 608 657 56 465 856 791 917 10 219 819 376 317 102 808 135 211 379 706 689 725 784 366 579 283 717 522 178 721 576 910 474 970 911 325 367 984 343 538 340 906 610 379 944 990 324 782 772 182 159 904 777 588 615 969 558 414 390 797 176 36 993 478 549 422 781 549 84 934 582 47 917 309 373 338 289 324 460 484 621 895 960 540 706 265 610 688 24 233 978 135 28 194 734 176 928 366 557 273 315 484 639 925 328 495 46 147 929 955 890 268 332 102 808 26 88 117 699 61 546 266 555 147 841 30 486 790 200 60 47 721 61 863 803 966 249 845 179 798 938 630 523 839 374 791 568 729 273 731 862 254 278 812 335 242 576 774 244 951 155 164 701 358 645 381 396 854 995 464 199 197 833 174 125 892 506 288 934 285 838 511 954 721 36 741 208 644 53 145 709 461 660 769 788 937 278 914 103 262 30 973 13 597 106 325 805 162 535 62 121 947 138 31 78 259 425 448 421 224 742 227 918 757 792 930 459 597 750 96 470 997 264 929 333 121 568 531 490 269 756 886 14 329 303 103 831 319 519 907 294 292 333 43 406 892 130 598 467 312 641 124 870 671 878 417 78 356 895 691 12 77 40 399 639 102 323 645 560 20 917 289 172 465 777 227 45 440 869 410 366 149 161 394 957 600 287 392 874 136 115

Pares celdas de página disponibles y fallos de páginas ocurridos:

160 767

200 741

240 719

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

280 111 624 597 358 333 335 725 158 400 823 165 877 326 311 373 848 451 834 412 802 696 15 643 464 250 572 692 246 765 2 219 332 62 567 275 785 974 162 739 944 159 246 850 4 500 68 803 53 975 146 217 218 463 748 855 847 674 307 645 906 539 940 707 449 934 855 811 619 580 908 193 558 978 683 181 885 833 803 127 823 580 842 85 406 585 805 74 160 127 265 251 739 918 429 985 995 808 119 871 402 58 219 272 149 658 755 5 417 688 827 31 74 454 912 905 578 122 823 810 55 933 800 709 634 87 775 618 768 816 653 923 341 971 875 311 536 847 274 47 240 425 401 863 875 12 711 931 829 416 521 988 947 4 158 415 269 582 715 845 112 494 786 959 467 72 808 95 364 963 17 197 577 980 129 921 363 474 11 650 554 513 700 698 641 896 125 180 383 167 154 684 330 716 944 625 620 778 454 520 485 901 48 41 438 621 794 723 780 360 784 512 206 74 427 582 95 482 711 967 220 248 820 903 697 643 557 854 177 878 377 153 161 178 877 665 226 227 85 676 640 680 425 487 568 67 916 348 795 602 1

857 218 771 701 763 915 668 463 385 571 985 160 636 579 744 456 721 473 529 995 506 500 568 732 686 316 680 20 831 703 911 431 610 647 242 330 738 161 155 254 599 828 372 632 890 249 691 357 64 125 229 472 659 820 748 897 777 430 967 966 877 194 988 943 990 57 495 172 136 191 446 120 777 674 337 782 485 609 64 762 410 779 704 556 151 755 22 100 446 342 746 514 993 386 627 741 758 980 73 391 156 129 919 480 451 220 6 156 864 929 669 467 600 753 593 455 256 996 971 988 935 346 720 126 320 583 592 985 170 580 279 317 166 840 591 798 586 624 30 430 157 191 948 185 727 417 911 427 951 763 829 851 218 268 612 721 194 595 461 50 6 782 55 434 406 147 182 416 322 150 575 558 982 792 918 346 798 509 202 658 645 34 492 255 490 232 569 490 850 368 611 366 438 479 909 917 815 972 888 941 785 954 473 401 545 54 336 366 884 994 254 800 126 611 83 117 381 565 472 604 287 408 251 265 821 300 107 516 130 103 531 621 151 786 494 580 727 564 902 601 584 709 740 622 697 776 286 378 959 216 875 466 231 978 872 216 358 262 252 757 205 394 252 736 163 383 159 272 602 542 743 982 364 529 6 412 407 703 332 172 375 390 906 481 489 888 225 190 323 550 625 848 214 300 223 439 103 788 419 357 319 296 807 986 749 951 362 438 722 419 482 898 602 660 8 60 583 584 557 311 632 198 231 961 817 28 14 577 415 495 484 199 278 815 169 558 351 677 999 338 691 122 411 618 195 368 224 521 481 15 278 897 67 777 930 498 995 190 608 26 302 998 819 254 754 216 586 850 608 244 669 64 795 8 835 19 407 225 279 897 42 108 1000 856 782 286 56 599 939 837 934 4 626 450 841 998 945 963 937 657 592 38 175 200 672 88 812 760 845 697 163 508 830 317 184 1 124 912 666 62 496 634 94 143 534 451 607 255 833 502 815 591 559 257 545 321 236 662 429 162 952 756 494 427 419 233 684 371 142 192 327 518 303 511 693 284 87 70 595 139 7 745 113 824 163 526 765 168 917 292 907 958 817 443 174 201 675 835 165 488 291 911 616 263 826 196 298 318 273 609 836 691 419 978 612 849 43 143 574 230 967 493 182 599 90 900 847 309 324 765 78 123 116 869 334 495 351 970 873 257 966 969 621 295 95 152 604 669 365 913 376 690 476 223 643 940 660 703 955 424 120 205 352 395 424 688 495 776 159 669 216 615 960 198 525 863 575 633 87 589 176 259 460 591 698 497 73 914 102 568 14 443 866 382 97 367 213 910 22 331 223 941 134 110 166 124 353 27 551 27 350 22 711 292 364 930 265 494 366 568 650 421 292 494 506 225 454 145 603 403 440 379 352 601 247 214 127 568 400 893 996 311 220 651 681 909 648 412 608 226 439 561 260 70 854 297 584 27 833 263 757 545 803 837 464 233 562 995 395

Pares celdas de página disponibles y fallos de páginas ocurridos:

160 772

200 749

240 722

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

749 359 994 706 779 545 779 794 548 560 752 180 836 79 11 132 388 942 968 451 459 499 45 885 952 636 585 69 16 23 646 875 42 873 716 238 575 526 241 359 95 369 741 801 338 642 202 232 358 787 402 958 62 514 122 484 371 261 83 97 38 840 323 757 317

314 660 520 262 928 666 383 209 702 997 204 503 193 216 835 451 144 263 285 558 561 731 4 932 415 962 415 326 545 714 373 41 121 17 55 580 478 759 16 479 299 666 345 203 36 939 388 507 837 530 70 649 295 614 860 651 7 364 261 220 251 297 509 44 70 37 847 24 494 801 183 849 504 331 373 142 391 999 507 295 348 484 854 235 584 508 928 184 1 481 212 393 331 433 927 524 93 837 833 690 647 237 946 780 555 598 769 777 588 279 67 347 316 290 671 430 562 626 128 785 740 475 451 645 911 959 168 690 374 121 213 196 221 182 945 560 367 706 550 892 619 328 529 369 277 437 883 929 722 905 604 629 994 220 205 794 898 445 144 321 997 805 481 889 963 625 193 563 988 483 546 627 321 830 77 549 411 927 841 887 932 89 13 275 377 160 940 285 688 651 277 158 471 384 177 59 568 714 801 85 933 184 184 647 686 13 22 700 432 236 860 90 882 556 157 864 204 359 784 534 537 353 547 687 886 839 775 316 349 799 22 853 280 822 203 719 471 439 746 550 273 776 952 10 618 466 168 519 691 657 331 665 216 917 835 927 107 50 362 991 926 655 88 209 37 966 622 512 140 136 148 755 783 56 919 448 350 384 294 376 13 125 390 147 791 361 231 348 782 848 109 627 989 837 827 951 116 315 966 957 512 970 619 832 837 604 631 124 217 111 597 935 658 198 793 550 117 359 698 921 202 498 166 53 688 654 442 106 43 915 655 834 982 195 450 566 682 314 289 993 392 675 476 631 166 698 505 131 630 792 435 382 137 844 659 552 518 848 305 182 887 733 576 650 212 185 972 262 263 360 143 43 599 716 518 297 716 578 501 853 216 122 992 980 201 251 445 612 510 365 576 975 154 803 885 939 61 869 743 305 272 869 775 806 437 212 887 525 959 30 884 690 794 951 175 666 697 627 329 542 793 40 329 361 796 224 536 763 303 93 946 135 569 399 474 228 257 917 300 570 203 781 609 216 204 988 377 324 565 478 679 688 401 445 606 229 91 417 370 332 243 984 940 852 528 17 644 814 144 773 439 870 147 263 932 302 294 963 501 321 506 955 877 722 182 518 169 676 468 924 549 559 488 56 868 704 567 388 143 495 624 913 751 432 20 6 987 661 882 766 861 685 217 948 823 334 45 185 59 263 72 115 909 26 677 655 740 453 794 2 598 64 532 71 565 584 188 640 710 915 470 554 29 720 349 716 743 478 485 221 163 15 924 157 486 217 71 608 105 363 413 513 947 145 530 870 530 200 927 266 119 470 379 971 516 167 296 247 281 350 282 870 833 696 133 625 717 67 676 477 156 257 771 687 738 194 835 563 706 574 691 491 837 496 314 286 763 759 443 208 791 737 133 332 85 615 611 331 958 96 74 385 360 111 703 548 413 274 33 237 982 955 683 374 295 46 532 465 944 893 682 801 90 867 643 552 525 397 243 501 859 279 935 886 100 858 894 244 459 360 619 187 440 568 18 186 849 318 136 891 792 889 340 739 407 624 570 420 656 312 494 519 387 487 422 261 357 433 423 839 244 270 990 91 277 556 598 861 989 445 340 710 511 247 890 757 278 406 795 444 618 213 185 281 497 354 450 528 874 528 380 905 399 271 724 326 753 600 212 731 610 431 449 998 571 560 160 757 894 24 572 656 584 308 452 737 505 851 315 661 923 16 389 454 937 558 175 311 233 585 627 914 944 870 691 694 596 331 329 58 74 474 908 428 987 550 688 225 111 927 470 210 44 973 171 404 294 94 578 247 420 533 581 410 252 983 876 916 353 162 498 744 765 974 540 452 744 558 210 412 124 337 456 762 505 305 971 702 700 934 240 752 207 628 125 905 1 445 929 429 864 655 708 614 591

Pares celdas de página disponibles y fallos de páginas ocurridos:

160 763

200 736

240 709

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

336 335 308 711 205 808 146 984 834 111 795 975 484 936 207 534 844 937 449 609 717 213 179 239 793 595 114 530 577 475 204 126 837 40 833 677 348 264 828 404 514 92 488 766 957 160 253 15 889 939 478 188 552 654 165 234 94 105 13 623 401 538 45 651 362 679 99 650 6 204 229 400 33 853 128 396 109 32 357 302 438 975 728 50 186 529 237 791 402 63 808 580 859 884 790 457 353 272 316 463 143 841 854 841 352 288 345 359 969 557 164 293 769 967 196 876 65 273 523 937 449 940 994 201 1 551 593 942 38 402 468 642 415 568 784 42 153 795 810 861 589 947 161 480 1 676 245 843 536 561 110 856 654 499 361 761 483 283 391 266 291 792 326 602 707 784 43 787 805 795 947 157 141 94 723 267 632 889 576 224 802 791 489 425 101 446 80 673 627 385 370 784 488 488 169 139 756 714 595 255 209 186 606 260 711 428 402 981 667 959 496 781 15 855 824 680 626 8 861 865 22 850 153 636 929 415 44 961 310 461 633 140 788 846 227 580 783 695 658 186 892 354 237 321 508 460 200 270 385 142 358 333 72 859 432 529 974 482 988 369 375 276 779 858 931 573 524 562 743 108 428 317 902 726 515 251 256 546 238 514 699 191 914 605 997 787 949 417 941 434 706 136 368 956 756 107 370 946 439 482 236 775 77 269 152 210 792 190 626 950 498 394 983 621 657 48 92 939 323 529 864 643 582 905 582 446 10 362 539 21 118 135 389 616 582 655 803 379 278 422 451 569 153 804 491 286 736 580 92 814 123 919 444 428 548 123 740 135 276 650 549 76 615 202 170 649 263 935 365 135 40 223 992 53 171 255 363 228 261 119 572 410 502 888 431 502 988 590 870 255 440 763 40 927 824 919 977 569 306 881 365 440 197 835 826 873 219 968 826 598 187 75 375 672 111 630 808 580 849 430 993 800 109 69 889 667 661 0 771 765 639 728 169 742 397 545 229 113 1 95 331 81 459 149 696 969 444 262 979 990 402 961 182 25 91 745 582 511 128 459 651 413 817 825 353 834 913 930 700 426 763 618 578 580 199 183 629 482 738 577 312 745 557 749 613 357 337 678 616 954 94 731 337 526 978 312 938 162 504 838 901 352 829 195 767 154 732 195 982 53 591 731 566 973 92 403 327 786 37 423 783 113 955 245 873 552 920 740 756 746 277 528 218 766 775 503 913 577 899 906 897 749 772 250 674 51 716 472 566 630 759 522 648 107 215 737 669 799 908 796 268 65 309 652 900 211 351 397 547 775 417 180 227 954 403 968 920 538 220 492 629 658 774 913 807 650 212 767 377 418 178 95 821 384 114 945 151 865 734 170 326 775 477 11 579 953 248 298 372 839 408 78 424 50 200 571 170 946 653 213 896 265 542 181 281 11 983 999 668 427 245 867 854 338 309 820 340 607 171 921 268 494 606 694 398 967 698 286 779 725 58 265 343 525 422 121 847 668 528 97 255 462 969 915 224 519 878 686 397 419 494 163 61 144 114 82 777 610 648 903 414 544 462 776 842 607 274 600 683 521 787 832 310 404 112 10 680 89 546 339 239 719 73 300 529 370 475 80 542 670 473 546 327 80 433 149 370 20 590 885 348 895 526 811 899 766 621 898 912 615 139 420 562 255 988 15 637 982 791 753 721 895 899 553 711 180 837 451 205 471 370 88 380 488 923 359 508 531 716 912 202 166 225 612 269 634 331 296 333 919 64 686 505 503 976 931 582 426 194 506 955 969 784 580 876 69 802 498 622 60 1 191 428 335 62 123 516 278 757 839 141 166 746 756 59 858 998 215 69 366 756 324 882 591 464 832 471 702 781 797 711 17 402 853 966 91 593 730 65 71 565 23 893 618 578 840 136 958 185 561 407

528 297 651 470 263 428 54 507 994 965 555 1 945 901 94 918 886 939 262 677 843 349 45 835 646 238 856 146 798 328 53 892 227 835 612 459 165 146 456 304 430 55 153 741 508 419 995 943 727 704 763 195 252 781 118 46 387 871 107

Pares celdas de página disponibles y fallos de páginas ocurridos:

160 769

200 750

240 730

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

658 983 823 502 827 213 287 329 36 699 963 536 157 598 217 885 452 436 392 903 334 378 226 374 498 546 279 246 458 384 689 413 423 581 137 451 806 878 910 127 808 70 316 540 292 279 65 128 951 446 838 12 926 269 918 802 348 260 247 231 890 883 435 543 662 872 306 98 342 541 352 347 209 33 453 572 579 127 868 830 715 275 562 369 736 146 111 592 884 82 552 225 733 507 186 500 676 673 553 938 438 998 172 499 839 775 587 618 241 783 518 264 144 779 606 804 520 792 675 457 536 993 765 609 727 379 54 289 833 741 603 773 762 490 802 916 111 527 176 176 567 29 482 838 446 148 165 528 77 420 139 631 641 321 421 353 251 96 327 226 625 539 804 217 612 722 957 227 602 251 353 460 101 689 84 378 900 67 853 519 522 866 526 970 573 879 221 499 517 480 147 699 150 660 858 894 363 391 229 89 471 410 44 223 168 462 119 179 826 606 801 613 322 97 509 370 885 52 545 470 359 968 81 963 837 915 328 693 551 223 447 427 353 678 812 232 198 339 984 87 901 371 542 129 264 618 81 932 516 331 780 671 856 289 863 769 704 771 72 505 995 712 551 772 936 633 714 714 117 466 854 470 596 130 281 337 551 664 690 392 715 130 793 931 739 439 309 331 540 461 332 889 818 802 573 785 967 992 735 550 981 228 652 359 431 199 204 986 967 196 717 52 394 188 340 208 703 896 270 218 603 496 397 301 422 286 876 141 852 298 360 231 727 888 982 123 241 939 475 28 788 465 820 623 784 249 376 27 633 307 453 229 157 931 142 521 439 677 872 112 175 134 918 50 953 334 310 190 593 458 819 989 412 729 98 237 205 714 228 992 314 589 736 280 620 197 621 960 119 618 449 587 308 512 841 618 215 779 5 122 472 36 900 416 183 204 766 5 815 817 386 64 113 236 569 190 230 70 899 276 102 743 944 874 432 985 70 42 334 935 327 831 235 251 304 8 987 800 514 69 908 291 431 911 308 403 502 588 405 992 181 388 285 413 180 3 398 598 213 893 37 604 221 974 189 866 896 74 846 505 728 443 572 750 525 984 249 552 583 315 332 446 543 136 19 193 74 328 291 531 915 237 935 527 243 373 647 518 299 405 734 317 735 393 495 340 12 8 484 389 240 457 464 883 398 166 586 925 403 551 404 18 975 807 781 321 424 475 372 305 265 946 916 737 637 920 43 209 715 573 326 719 477 807 102 74 154 298 723 11 167 623 162 320 322 885 531 565 217 858 977 233 657 117 146 73 756 941 131 598 167 272 106 187 134 979 19 60 765 653 784 148 778 697 145 85 41 717 909 223 16 971 243 264 672 291 311 712 977 314 322 65 71 346 390 418 913 327 132 986 424 794 448 702 861 478 380 343 169 609 773 433 454 220 835 363 73 434 28 165 605 496 454 272 233 231 900 920 484 86 485 570 761 599 819 651 750 173 430 691 412 504 911 556 768 20 599 746 677 358 523

524 404 643 320 937 663 47 67 364 151 558 970 655 468 968 497 260 841 857 477 380 214 314 773 304 900 270 632 430 384 888 846 24 773 291 744 912 660 816 646 122 192 264 157 332 636 43 305 26 72 670 109 12 809 378 645 162 951 365 801 194 330 935 967 741 478 351 31 485 657 780 469 438 67 462 305 859 26 489 584 320 316 899 499 509 53 898 992 745 83 694 24 270 697 78 873 627 775 290 526 748 356 454 399 426 729 133 282 950 635 261 230 695 608 229 747 446 500 635 395 395 146 876 822 142 451 938 523 776 227 677 484 804 14 13 714 411 569 65 367 907 504 418 442 361 407 715 954 491 746 675 945 108 106 223 679 608 673 239 662 307 280 527 105 165 938 289 742 500 148 830 162 341 953 826 625 286 988 567 224 862 575 421 979 961 812 482 942 930 989 556 617 737 734 255 634 531 604 464 198 406 485 832 149 400 633 434 691 96 754 511 144 218 255 45 257 126 70 783 536 438 265 992 760 926 752 989 76 413 70 788 633 352 136 894 531 924 383 580

Pares celdas de página disponibles y fallos de páginas ocurridos:

160 777

200 745

240 713

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Vector de requerimientos de páginas para la simulación:

314 49 201 904 960 607 848 218 210 590 829 482 142 779 426 41 126 112 327 70 819 439 213 377 105 355 224 41 798 746 943 129 873 244 537 732 965 243 184 362 476 861 772 761 195 86 687 45 94 734 913 956 59 426 42 781 80 988 237 304 938 990 390 865 596 107 550 403 322 672 706 879 427 202 315 594 572 324 954 594 45 665 127 635 367 212 693 321 619 467 958 49 632 221 147 337 554 376 60 315 548 321 924 744 413 763 356 477 704 802 238 241 107 702 671 790 430 176 412 844 653 673 914 782 187 833 768 302 235 195 257 28 255 129 44 430 696 981 140 516 457 565 378 893 747 153 931 614 558 116 212 917 655 658 445 861 860 977 776 601 893 241 243 214 109 58 970 768 536 752 762 450 414 603 916 73 789 206 419 3 338 686 68 883 269 796 436 850 265 832 282 945 948 388 28 286 544 213 438 818 506 724 544 11 156 976 787 69 706 638 651 284 542 516 169 667 122 909 693 168 309 85 659 53 762 802 743 56 629 506 394 597 185 961 949 707 99 973 886 183 606 826 331 755 405 360 249 150 86 283 603 357 148 674 842 570 137 943 675 185 321 118 734 949 426 956 323 34 410 978 165 519 735 879 47 813 506 637 709 350 365 562 383 462 645 810 968 504 588 503 320 184 584 64 666 342 37 952 255 806 91 888 814 304 94 794 744 104 708 168 703 683 752 480 332 175 440 281 63 971 590 337 434 744 496 201 439 422 601 456 496 488 65 358 62 982 388 225 279 504 707 494 350 467 265 922 607 209 885 511 933 753 702 514 532 425 456 320 723 203 656 20 661 904 413 491 40 546 259 100 558 492 170 520 770 197 58 366 667 43 464 956 817 532 629 169 990 723 860 101 136 715 113 649 223 999 20 349 608 614 626 352 482 346 786 460 272 676 77 907 593 658 961 585 777 28 324 744 860 833 931 317 778 809 579 774 290 536 406 503 213 312 178 784 625 255 715 608 471 929 513 87 57 882 637 109 651 515 184 462 287 300 888 347 384 117 595 462 161 739 285 739 572 626 818 464 939 622 830 300 256 707 525 495 9 210 844

558 610 324 273 187 774 809 146 480 648 823 293 358 714 10 54 788 176 581 55 103 293 410 513 25 913 20 174 421 992 638 922 875 216 168 206 57 143 664 99 874 288 994 188 938 424 271 623 719 697 727 180 674 969 800 837 313 354 622 28 710 492 334 26 30 319 625 206 984 703 104 889 562 532 204 677 75 818 256 861 816 787 114 306 900 773 678 796 201 70 832 913 24 748 478 470 713 838 9 280 23 686 165 941 34 689 338 20 204 125 574 174 106 805 499 581 939 447 577 996 418 172 54 321 975 881 616 404 441 255 882 688 434 720 348 432 919 75 533 454 174 282 198 726 103 908 702 788 931 37 334 5 831 528 334 761 579 322 926 895 959 644 32 94 122 442 88 306 857 40 987 694 828 924 537 18 82 14 324 303 552 913 131 606 532 141 666 492 876 843 352 510 618 403 602 401 63 442 470 524 152 197 372 966 927 464 911 945 441 904 105 991 915 477 66 900 12 56 227 117 198 560 348 515 454 846 100 986 516 992 872 934 765 199 193 589 311 413 988 891 292 643 864 240 791 951 274 188 513 908 469 90 719 769 87 470 935 101 8 817 417 678 211 474 187 147 667 24 90 787 55 64 916 806 547 98 867 704 73 65 352 855 589 445 510 468 738 26 494 612 59 416 838 58 701 437 531 244 409 306 992 6 171 864 3 322 241 410 596 481 177 148 538 964 582 694 580 571 115 185 958 665 151 328 894 952 265 666 424 309 93 585 454 834 226 811 565 768 810 356 111 380 459 255 220 559 933 575 955 981 324 905 697 961 193 851 523 581 899 780 828 132 302 555 110 627 763 851 718 124 478 207 374 981 703 226 976 337 287 202 716 834 5 931 9 18 951 86 129 680 871 648 271 396 751 917 394 63 129 502 105 463 900 432 601 605 633 991 56 910 139 946 366 425 505 301 992 229 776 644 405 962 219

Pares celdas de página disponibles y fallos de páginas ocurridos:

80 830

100 813

120 801

NO se ha producido la Anomalía de Belady en la simulación efectuada.

Bibliografía

[1] L. Joyanes Aguilar. *Fundamentos de Programación - Algoritmos y Estructura de Datos - Segunda Edición.* Mc Graw Hill/Interamericana de España, S.A.U., España, 1996.

[2] L. Joyanes Aguilar. *Programación Orientada a Objetos - Segunda Edición.* Mc Graw Hill/Interamericana de España, S.A.U., España, 1998.

[3] L. Joyanes Aguilar; L. Rodríguez Baena; M. Fernández Azuela. *Fundamentos de Programación - Libro de Problemas.* Mc Graw Hill/Interamericana de España, S.A.U., España, 1996.

[4] J. Boria. *Construcción de Sistemas Operativos.* Kapelusz, Bs.As.-Argentina, 1987.

[5] E. Castillo; A. Iglesias; J. M. Gutiérrez; E. Alvarez; A. Cobo. *Mathematica.* Paraninfo, España, 1996.

[6] E. Castillo; M. Reyes Ruiz Cobo. *Preparación de Documentos con Latex.* Maestría en Informática y Computación - FACENA - UNNE, Argentina, 1998.

[7] H. M. Deitel. *Introducción a los Sistemas Operativos.* Addison-Wesley Iberoamericana, México, 1987.

[8] W. H. Press; S. A. Teukolsky; W. T. Vetterling; B. P. Flannery. *Numerical Recipes in C, second Edition.* Cambridge University Press, Cambridge, 1992.

[9] E. Castillo; J. M. Gutiérrez; A. S. Hadi. *Sistemas Expertos y Modelos de Redes Probabilísticas.* Academia de Ingeniería, España, 1996.

[10] J. R. Hilera González; V. J. Martínez Hernando. *Redes Neuronales Artificiales - Fundamentos, Modelos y Aplicaciones.* Addison-Wesley Iberoamericana, Delaware-USA, 1995.

[11] A. Iglesias. *Curso Avanzado de Métodos Numéricos.* Corrientes, Argentina, 1998.

[12] Borland International. *Turbo C++ - Getting Started.* Borland International, USA, 1990.

[13] A. Kvitca. *Resolución de Problemas con Inteligencia Artificial.* EBAI, Brasil, 1988.

[14] G. D. Pino; L. A. Marrone. *Arquitecturas RISC.* Kapelusz, Bs.As.-Argentina, 1987.

[15] L. Joyanes Aguilar; I. Zahonero Martínez. *Estructura de Datos - Algoritmos, Abstracción y Objetos.* Mc Graw Hill/Interamericana de España, S.A.U., España, 1998.

[16] R. Penrose. *La Nueva Mente del Emperador.* Mondadori España S. A., España, 1991.

[17] E. Castillo; A. Cobo; J. M. Gutiérrez; R. E. Pruneda. *Introducción a las Redes Funcionales con Aplicaciones - Un Nuevo Paradigma Neuronal.* Paraninfo, España, 1999.

[18] A. C. Shaw. *The Logical Design Of Operating Systems.* Prentice Hall, NJ-USA, 1974.

[19] J. L. Peterson; A. Silberschatz. *Operating Systems Concepts.* Addison-Wesley, MA-USA, 1991.

[20] E. Castillo; A. Cobo; P. Gómez; C. Solares. *JAVA - Un Lenguaje de Programación Multiplataforma para Internet.* Paraninfo, España, 1997.

[21] W. Stallings. *Data and Computer Communications - Fifth Edition.* Prentice Hall, NJ-USA, 1997.

[22] A. S. Tanenbaum. *Operating Systems: Design And Implementation.* Prentice Hall, NJ-USA, 1987.

[23] A. S. Tanenbaum. *Sistemas Operativos Modernos.* Prentice Hall Hispanoamericana, S.A., México, 1993.

[24] A. S. Tanenbaum. *Organización de Computadoras - Un Enfoque Estructurado - Tercera Edición.* Prentice Hall Hispanoamericana S. A., México, 1996.

[25] A. S. Tanenbaum. *Sistemas Operativos Distribuidos.* Prentice Hall Hispanoamericana, S.A., México, 1996.

[26] A. S. Tanenbaum. *Redes de Computadoras.* Prentice Hall Hispanoamericana S. A., México, 1997.

[27] M. L. James; G. M. Smith; J. C. Wolford. *Métodos Numéricos Aplicados a la Computación Digital.* Departamento de Ingeniería Mecánica, Universidad de Nebraska, Representaciones y Servicios de Ingeniería, S.A., International Textbook Company, México, 1973.

Índice de Materias